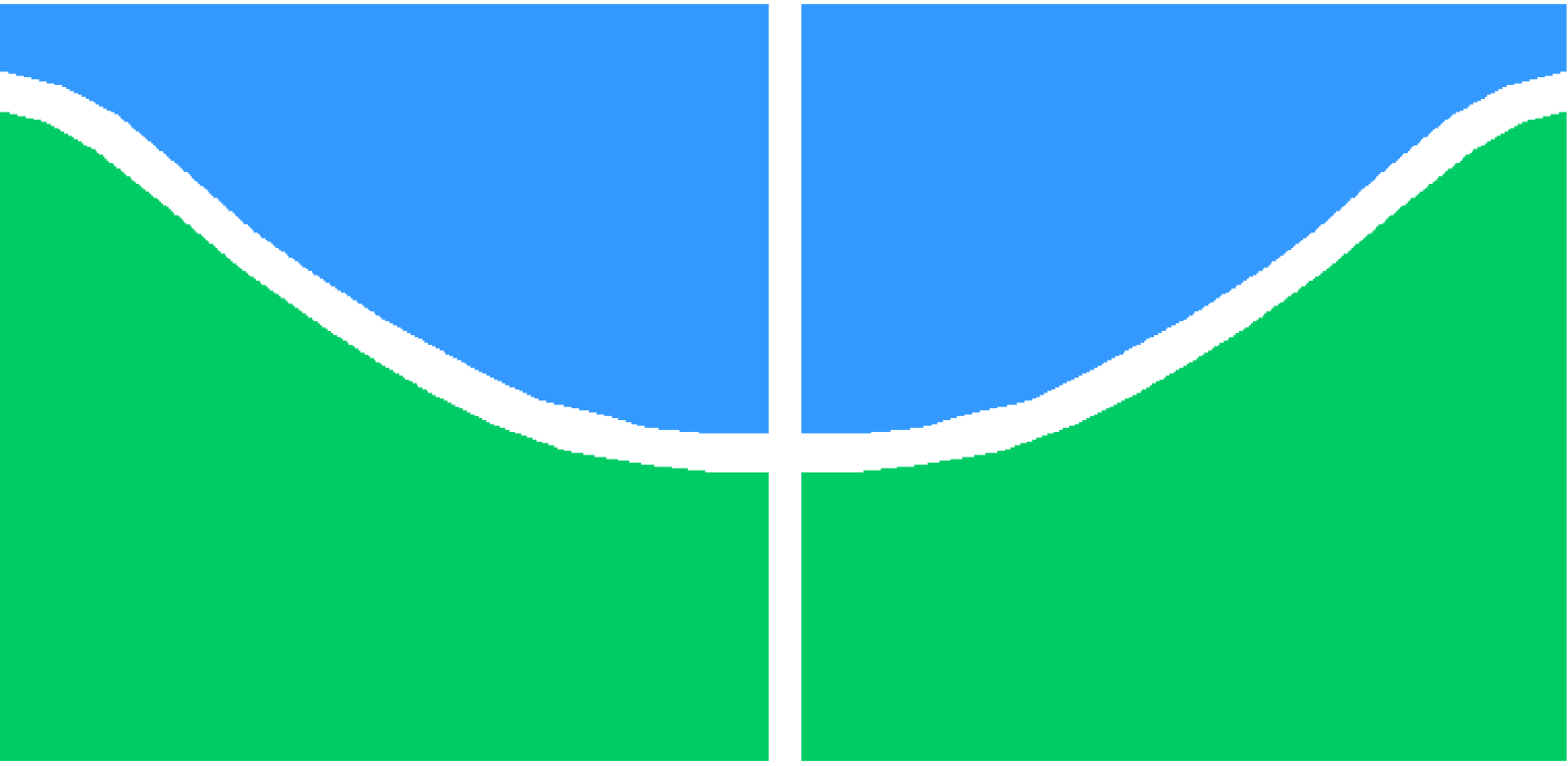

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de Software

Audiobooks com navegação por marcações de conteúdo como ferramenta de apoio didático para não-videntes

Autor: Bryan de Holanda Fernandes

Orientador: Prof. Dr. Carla Rocha

Brasília, DF

2016

Bryan de Holanda Fernandes

Audiobooks com navegação por marcações de conteúdo como ferramenta de apoio didático para não-videntes

Monografia submetida ao curso de graduação em *Engenharia de Software* da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. Dr. Carla Rocha

Brasília, DF

2016

Bryan de Holanda Fernandes

Audiobooks com navegação por marcações de conteúdo como ferramenta de apoio didático para não-videntes/ Bryan de Holanda Fernandes. – Brasília, DF, 2016-

97 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Carla Rocha

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2016.

1. Não-videntes. 2. Audiobooks. I. Prof. Dr. Carla Rocha. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Audiobooks com navegação por marcações de conteúdo como ferramenta de apoio didático para não-videntes

CDU 02:141:005.6

Bryan de Holanda Fernandes

Audiobooks com navegação por marcações de conteúdo como ferramenta de apoio didático para não-videntes

Monografia submetida ao curso de graduação em *Engenharia de Software* da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 14 de dezembro de 2018:

Prof. Dr. Carla Rocha
Orientador

Convidado 1

Convidado 2

Brasília, DF
2016

Agradecimentos

A Deus acima de tudo e por tudo que tem feito em minha vida. Pelas oportunidades que Ele tem concedido a mim. Agradeço a minha família e amigos pelo apoio, carinho e consideração.

"Nada façais por contenda ou por vanglória,
mas por humildade;
cada um considere os outos superiores a si mesmo.
Não atente cada um para o que é propriamente seu,
mas cada qual também para o que é dos outros.
(Bíblia Sagrada, Filipenses 2, 3-4)

Resumo

O *audiobook* é um recurso informacional, seja ele no campo científico, social, educacional e político, que tem promovido a inclusão social de pessoas no acesso a informação e ao conhecimento. Através de meios eletrônicos, eles também tem contribuído com a sociedade proporcionando maior acessibilidade, agilidade na busca da informação, compartilhamento, dentre outros fatores. Os deficientes visuais tem tido uma participação cada vez maior junto às novas tecnologias da informação, visto que, pessoas que possuem a limitação visual, o acesso a informação acaba sendo restrito. O *audiobook*, como um recurso informacional gerado pelas novas tecnologias, pode contribuir para o aprimoramento educacional e desenvolvimento crítico através da leitura às pessoas com deficiência visual. Este recurso pode também ser utilizado por pessoas que não possuem deficiência visual, no que tange a cooperação com a formação e regate de leitores como também na utilização para instrução e aprendizado de uma segunda língua e suporte a alfabetização. Para melhor utilização e tornar acessível para um número maior de pessoas, um novo formato foi especificado oferecendo suporte a navegação do conteúdo do *audiobook*. O formato faz uso da compressão de dados por considerar o compartilhamento e um player que dê suporte ao novo formato também foi desenvolvido.

Palavras-chaves: Cego. Pessoa com deficiência visual. Cegueira. Baixa visão. *Audiobooks*. Marcação de conteúdo. Ogg Vorbis. *Open Source*.

Abstract

The audiobook is an informational resource, be it in scientific, social, educational and political, that has promoted social inclusion of people in accessing information and knowledge. Through electronic means, they have also contributed to society by providing greater accessibility, agility in search of information, sharing, among other factors. The visually impaired has had an increasing participation with ace new information technologies, since people who have a visual impairment, access to information ends up being restricted. The audiobook as an informational resource created by modern technology can contribute to educational improvement and critical development by reading to non-seers. This feature can also be used by people who are blind, when it comes to cooperation with the training and regattas players as well as the use for instruction and learning a second language and literacy support. To better use and make it accessible to a larger number of people, a new format was specified oferendo support the navigation of audiobook content. The format makes use of data compression considering sharing and player that supports the new format has also been developed.

Key-words: *Audiobooks*. Bookmarking Content. Ogg Vorbis. *Open Source*.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

As pessoas estão mais disponíveis ao uso dos dispositivos eletrônicos e isto tem provocado um afastamento das versões impressas e uma consequente aproximação de livros eletrônicos. Os *e-books*, nome dado aos livros eletrônicos, estão fazendo sucesso e sendo cada vez mais utilizados pelos editores para tornar disponíveis livros em formato eletrônico. Se comparado aos livros impressos, os *e-books* oferecem diversos benefícios aos leitores, incluindo portabilidade, redução de danos ao meio ambiente e menor peso (BURKEY, 2013).

Para os deficientes visuais, estes livros eletrônicos oferecem algo ainda mais importante: a sua primeira oportunidade de desfrutar da leitura. Seja um cidadão cego ou com baixa visão, seja uma pessoa com dislexia ou outras dificuldades de aprendizagem, seja uma criança que não consegue segurar ou mudar a página de um livro, todos eles encontram enormes dificuldades para ler livros impressos. Dependendo do grau do campo de visão de uma pessoa, mesmo com o *e-book* a leitura ainda é impossível (BLANCK, 2010).

Algumas das limitações citadas aparentemente não poderiam ser solucionadas através da produção dos livros eletrônicos. No entanto, os *e-books* tem potencial para alcançar cerca de 30 milhões de pessoas com deficiência, pois embora sejam interpretados por um computador na maioria dos casos, eles não estão restritos a ele, e tão pouco estão limitados ao formato de texto. Ou seja, os *e-books* podem ser processados em texto eletrônico, Braille, hieróglifos, bem como na própria impressão em papel. Não somente a isso, os *e-books* também podem ser apresentados em formato acústico, recebendo o nome de *audiobooks* (BLANCK, 2010).

Os *audiobooks* tem crescido no mercado mundial. Eles são amplamente difundidos pela internet, e em sua maioria no formato MP3. Este formato tem sido difundido por armazenar áudio de alta qualidade e ocupar pouco espaço em memória, devido ao uso de uma técnica de compressão de dados. Como desvantagem, quaisquer tipos de anotações e marcações a serem acrescidas ao *audiobook* neste formato são ligadas diretamente ao tocador e não ao arquivo.

Outro ponto importante é a não existência de formatos *open source* que ofereçam suporte para anotações e marcações lógicas. Estas marcações devem permitir navegar pela estrutura do conteúdo do livro (por exemplo, saltar entre capítulos, parágrafos, etc), e os que possuem são protegidos por leis de propriedade intelectual. Chamadas neste trabalho de marcações de conteúdo, estas referências de posição representam a estrutura lógica de um livro físico, isto é, capítulos, seções, parágrafos, versículos, entre outros (REIS,

2013). Reis especificou um formato aberto para *audiobooks* com suporte a marcadores de conteúdo.

Este trabalho visa evoluir a especificação do formato proposto por Reis de modo a obter um formato de arquivo para *audiobooks* capaz de conter o áudio, os metadados e marcações de conteúdo referente ao áudio comprimido, bem como desenvolver um player capaz de executar tal formato de forma acessível, através da interação pelo teclado. O desenvolvimento do formato e da ferramenta tem como intuito investigar a seguinte hipótese de pesquisa: *audiobooks* com navegação acessível por marcações de conteúdo podem contribuir com o ensino para pessoas com deficiência visual que possuem limitações para ler livros didáticos.

1.1 Objetivos

Nesta seção serão descritos os objetivos que se pretende alcançar com este trabalho e estão divididos em objetivo geral e objetivos específicos.

Objetivo Geral: especificar um arquivo de audiobook com compressão que suporte a navegação e testar em campo a hipótese de que um audiobook com navegação por marcações de conteúdo podem ser utilizados como ferramenta didática para cegos ou pessoas com baixa visão;

Objetivos Específico 1: especificar um arquivo de áudio compactado. Alguns critérios devem ser contemplados, tais como fazer uso da compressão de dados, ser capaz de armazenar conteúdo para suporte a marcação do conteúdo do áudio, conter informações referente ao áudio armazenado e ser um formato open source;

Objetivos Específico 2: Desenvolver um tocador para tal formato com recurso a acessibilidade para pessoas com deficiência visual. Para tanto, o player deve ser capaz de decodificar e executar o novo formato especificado sem interrupção ou perda de informação trabalhando com todas as informações contidas no formato. Pular para pontos específicos do áudio também é um requisito que o tocador deve possuir com interação por meio do teclado para dar suporte a utilização da ferramenta aos cegos ou pessoas com baixa visão;

Objetivos Específico 3: elicitar requisitos que são acessíveis para cegos e pessoas com baixa visão necessários a utilização do Tocador;

Objetivos Específico 4: analisar os requisitos elicitados e realizar estudo de viabilidade para implementação das novas funcionalidades propostas;

Objetivos Específico 5: descrever os requisitos viáveis ao projetos em *User Stories* e documento de Casos de Uso; e

Objetivos Específico 6: implementar os requisitos descritos no Tocador desenvolvido.

1.2 Motivação

Apesar do avanço tecnológico e o surgimento e popularidade dos *audiobooks*, ainda existem barreiras para se ter acesso a leitura. De um modo geral, isso não se dá ao fato da limitação da tecnologia, mas ao foco que as pessoas estão dando. A diretriz que se tem seguido é tornar tudo mais fácil e prático para quem já faz uso. Com esta linha de pensamento, acaba-se por esquecer em tornar algo acessível para um grupo específico de pessoas que não teve ainda uma oportunidade de uso e costume da prática da leitura. Viabilizar a acessibilidade focando na inclusão social de um grupo específico de pessoas e dando a elas uma oportunidade de leitura é uma das principais motivações deste trabalho. Pretende-se maximizar o ensino e ajudar as pessoas com deficiência visual a atingirem seu potencial e a fazerem a sua plena contribuição a sociedade.

1.3 Delimitação do Assunto

A gravação de um livro falado é, em suma, realizada no formato MP3 principalmente por ser um dos primeiros formatos por fazer uso da compressão sem perdas dados e por ser o formato mais disseminado nas redes e compartilhamento. No entanto, o formato MP3 é patenteado e não oferece suporte para os audiobooks. Devido a dificuldade ao acesso a informação que não se aplica apenas aos cegos ou pessoas com baixa visão, pois mesmo para as pessoas que não possuem nenhum tipo de limitação visual ou motora, navegar por horas contínuas de áudio em busca de informação é trabalhoso e em sua maioria impraticável. Um novo formato deve ser especificado e um player que suporte este formato deve ser também desenvolvido para que seja possível obter informações referentes a gravação e marcações de conteúdo que possiblite a navegação facilitando o acesso a informação.

O novo formato foi derivado do formato Ogg Vorbis por possuir os requisitos básicos necessários para o desenvolvimento deste trabalho. O primeiro deles é que o Ogg Vorgis já oferece suporte a compressão de dados. Como o foco deste trabalho não é a compressão de dados, ela foi feita através de rotinas da libvorbis, pois é uma biblioteca open source, bem consolidada e testada. O formato também possui uma estrutura que facilita a inserção de novos pacotes de dados que não ocasionará na quebra de sua estrutura. Outro requisito é o livre acesso para o uso e desenvolvimento do formato, pois é open source. O player desenvolvido foi construído para suportar o formato especificado e promover um melhor uso pelas pessoas com deficiência visual, pois é o foco deste trabalho. Assim sendo, o player possui interação por teclado, suporte para navegação pelo conteúdo do arquivo, interface com ampliador de tela e contraste e sintetizador de voz.

1.4 Organização do Trabalho

O trabalho está organizado em capítulos. No Capítulo 2 é apresentada a revisão bibliográfica que forneceu o embasamento teórico para o entendimento e desenvolvimento do projeto.

No Capítulo 3 são apresentadas todas as etapas realizadas para a especificação e evolução do formato, e para o desenvolvimento do Editor e do Tocador, bem como as ferramentas utilizadas no processo de desenvolvimento e pesquisa. No Capítulo 4 são apresentados os resultados obtidos e, por fim, no Capítulo 5, será apresentado a conclusão do trabalho realizado e diretrizes para sua melhoria e possíveis avanços.

2 Fundamentação Teórica

Neste capítulo estão as pesquisas e todo o embasamento teórico referente a literatura acerca de trabalhos já desenvolvidos que dão suporte e direcionam o tema deste trabalho. As seções contidas neste capítulo abordam temas referentes ao surgimento e crescimento dos *audiobooks* bem como os formatos existentes, acessibilidade, pessoas com deficiência visual e suas dificuldades e limitações, formato de áudio e também traz uma contextualização sobre a viabilidade e aceitação do tema proposto.

2.1 Terminologia

Segundo Romeu Kasumi Sassaki, "a construção de uma verdadeira sociedade inclusiva passa também pelo cuidado com a linguagem" (SASSAKI, 2005). Em seu estudo, Sassaki relata sobre a importância da linguagem utilizada pois através dela podemos expressar, seja voluntariamente ou não, o respeito ou a discriminação em relação às pessoas com deficiência. Dentre as terminologias corretas citadas por Sassaki nós podemos utilizar: cego, pessoa cega, pessoa com deficiência visual, deficiente visual, cegueira e baixa visão. A deficiência visual diferencia-se entre deficiência visual parcial, quando a pessoa possui baixa visão, e cegueira, quando a deficiência visual é total (SASSAKI, 2005).

O termo portador de deficiência foi largamente utilizado no Brasil entre os anos de 1986 e 1996. No entanto, em 2003, Sassaki disse que o uso é incorreto visto que a deficiência não é uma coisa que portamos ou não como um documento de identidade, por exemplo, e sugere a terminologia pessoa com deficiência como correto (SASSAKI, 2005). Em 2006, a Assembleia da ONU aprovou a Convenção Sobre os Direitos das Pessoas com Deficiência onde, assinada pelo Brasil em 2007 e ratificada pelo Congresso Nacional em 2008, acaba por oficializar o termo pessoa com deficiência em seu próprio título (LEGAL, 2006).

A Câmara dos Deputados utiliza e orienta as pessoas a utilizarem o termo hoje mundialmente aceito, a saber: pessoa com deficiência. Diz também que os termos cego e surdo podem ser utilizados (DEPUTADOS,).

2.2 Audiobooks

Podemos dizer que *audiobook* é a versão em áudio de livros impressos. Também conhecido como "livro falado", o *audiobook* tem sido conhecido aos poucos aqui no Brasil e utilizado como suporte para que pessoas que não possuem o hábito de ler passam a

conhecer obras literárias. A partir da leitura do livro feita em voz alta, o conteúdo do livro é gravado e armazenado em formatos diversificados tais como CD, MP3, WMA e Ogg. O controle de qualidade destas gravações estão ligadas principalmente ao formato que está sendo utilizado como suporte. Outros requisitos, não menos importantes, tratam dos narradores e da interpretação que se dá ao personagem, das variações no tom de voz em diálogos bem como os efeitos sonoros produzidos trazendo maior realismo e um maior envolvimento na história ajudando na interpretação do texto (PALMER, 2013).

O audiobook é uma forma inovadora de acesso à leitura (SOUZA; CELVA; HELVADJIAN, 2006). Mesmo para aquelas pessoas que possuem o hábito de leitura, este hábito vem se perdendo devido ao ritmo acelerado das grandes cidades do mundo de hoje. Por falta de tempo, as pessoas tem feito uso de *audiobooks* para estarem "lendo" enquanto passam horas dirigindo em extensos engarrafamentos ou enquanto fazem exercícios físicos (PALLETA; WATANABE; PENILHA, 2008). Isso é possível pois a portabilidade do audiobook em comparação aos livros impressos é muito maior, pois o local não parece importar. Os audiobooks também podem ser utilizados em sala de aula como suporte à alfabetizacão e como instrução de uma segunda língua.

Desde o início da década de 50, a apreciação pela literatura falada vem crescendo e se tornando tradição nos Estados Unidos. Mas foi em 1980 que realmente os audiobooks ganharam corpo. Segundo (TEIXEIRA, 2006) havia, em 2006, cerca de 30 mil títulos de *audiobooks* nos Estados Unidos disponíveis considerando apenas o site da Audible, companhia da Amazon. No ano de 2014, a companhia possui mais de 150 mil títulos disponíveis para download (AUDIBLE, 2014). Este número mostra como o mercado de *audiobooks* vem crescendo no mercado norte-americano.

O crescimento do uso de *audibooks* não tem ocorrido apenas nos Estados Unidos. Na Europa, os livros falados também tem ganhado força, sendo mais popular na Alemanha. Neste país, o hábito virou moda na década de 90. Podemos citar o Instituto Goethe que é um instituição sem fins lucrativos que visa a disseminação do idioma e cultura alemã. Eles possuem *streaming* de diversas obras literárias alemãs hospedadas e disponíveis em seu site. Sabe-se que dois eventos populares na Alemanha, a grande Feira Internacional de Livros de Leipzig e o Festival Internacional de Literatura de Berlim, contam com estandes de venda voltados à apresentação de *audiobooks* (DW, 2004).

Oscar Niemeyer, antes mesmo do uso dos CDs, certa vez disse "Quando quero ler, eu ouço. Pago uma pessoa para gravar os livros em fitas e depois, quando sinto vontade, as coloco para tocar" (PALLETA; WATANABE; PENILHA, 2008). Aqui no Brasil, o uso de *audiobooks* começou um pouco mais tarde e em um ritmo um pouco mais lento se comparado com Estados Unidos e Europa. No entanto, nos últimos anos os livros falados tem sido cada vez mais utilizados pela população brasileira. O mercado brasileiro tem correspondido a esta nova disseminação cultural (FARIAS, 2012). Isto mostra que

a cultura de livros falados no Brasil veio para ficar. Plugme e Audiolivro Editora são editoras que tem investido em audiolivros no Brasil. Em 2008, quando o mercado ainda era considerado tímido, a Plugme registrou uma venda de setecentos livros por mês contra mil livros da Audiolivro (SOUZA; CELVA; HELVADJIAN, 2006). A Audiolivro conta com mais de 900 títulos. Voltado para a literatura infantil, a editora RHJ reúne mais de 200 publicações e premiações recebidas no exterior tais como *White RAvens* na Alemanha, *Octogone* na França, *BIB Plaque* na Eslováquia, *The Noma Concours* no Japão e *The Ibby Honour List Diploma* no Canadá. A editora RHJ também possui premiações no Brasil (RHJ, 2009).

A Universidade Falada é um portal de iniciativa privada que visa difundir cultura pelo Brasil com distribuição de conteúdo em áudio viabilizado pela *Editora Alyá* com preços acessíveis e facilidade para a aquisição. Este projeto conta com mais de mil e trezentos audiolivros e estimam mais de cinco mil horas de áudio disponíveis em formato mp3. (FALADA, 2004).

> O projeto Universidade Falada® pretende democratizar a cultura. Facilitar o acesso, em formato áudio e audiolivro, a grandes obras da literatura nacional e internacional à população mais afastada dos grandes centros culturais do país. Nossa missão é ajudar pessoas, oferecer conhecimento e cultura. Discutir temas velhos e novos, ensinar e filosofar. Agregar valor ao ser humano. É isso que nós editores, autores e palestrantes desejamos desta empreitada. Nossos preços permitem que qualquer brasileiro com acesso a internet possa adquirir nossos produtos. Sem exceção (FALADA, 2004).

Em outubro deste ano, a *PublishNews* publicou uma matéria informando a chegada de duas plataformas de audiolivros para smartphones e tablets no mercado brasileiro. A UBook é uma delas e já está disponível para plataforma iOS e Android. Adotando o serviço de subscrição, os usuários possuem acesso ilimitado a mais de mil obras. Segundo os desenvolvedores, cinco dias após seu lançamento, a plataforma já possuia mais de vinte mil usuários. Uma outra alternativa é a TocaLivros que optou pela venda unitária do audiolivro em formato digital (NETO, 2014).

2.3 Audiobook e as Pessoas com Deficiência Visual

Como estrelas que brilham no campo editorial é evidente a popularidade dos audiobooks (BURKEY, 2013). No entanto, acessibilidade ainda é um desafio para os publicadores pois os formatos não são totalmente acessíveis (SHAFFI; WOOD, 2014). Um jornal televisivo de média audiência levou ao ar uma matéria que trata da experiência de pessoas com deficiência visual no uso da Internet. Um dos temas tratados foi sobre softwares que fazem leitura de tela. Uma voz robotizada permite ao usuário cego ouvir a informação que

está sendo apresentada para ele na tela do computador. Porém, dependendo da estruturação da página que se está acessando, a pessoa com deficiência visual é incapaz de acessar a informação ou o processo é muito demorado. Isso ocorre porque os elementos da página web podem estar dispostos de tal forma que o deficiente visual levará um longo tempo até acessar determinado link ou chegar a um um texto/tópico de seu interesse. Ou seja, as informações dispostas na página web não estão agrupadas ou categorizadas de forma a tornar o acesso a informação mais eficiente pelo uso do software. Ao invés disso, cada link, coluna, texto, título, subtítulo, notas de rodapé, serão lidos pelo software um após o outro. Por meio desta matéria, podemos facilmente observar que apesar da possibilidade de se ouvir um livro, o acesso a informação para uma pessoa que possui deficiência visual ainda é restrito e limitado. Buscar uma informação em meio a contínuas e extensas horas de gravação ou áudio é um trabalho custoso e demorado.

Troy Juliar é diretor de conteúdo da Recorded Books, uma editora de áudio global com catálogo de mais de 24 mil títulos e um programa de publicação anual de mais de 3 mil títulos. Juliar trabalha na área há mais de 25 anos e quando questionado se o desenvolvimento de novos formatos de áudio como suporte entre as pessoas com deficiência visual e os audiobooks são importantes a resposta é: absolutamente.

> Como uma indústria, temos aumentado significativamente a produção ao longo dos últimos anos - todos nós estamos trabalhando muito duro para produzir o máximo possível de forma tão eficiente quanto possível, mantendo a qualidade. Em cima disso, nós queremos aumentar a conscientização sobre o formato - uma tarefa muito importante no mundo de hoje de muitas opções concorrentes (MALCZEWSKI, 2012).

Os formatos existente não são totalmente acessíveis e, para torná-lo ainda mais acessível, funcionalidades que incluam marcações de conteúdo devem ser pensados (SHAFFI; WOOD, 2014).

2.4 Pessoas com Deficiência Visual

Conforme o Decreto nº 3.298/99 e o Decreto nº 5.296/04, conceitua-se como deficiência visual:

- cegueira - na qual a acuidade visual é igual ou menor que 0,05 no melhor olho, com a melhor correção óptica;
- baixa visão - significa acuidade visual entre 0,3 e 0,05 no melhor olho, com a melhor correção óptica;
- Os casos nos quais a somatória da medida do campo visual em ambos os olhos for igual ou menor que 60°;

- Ou a ocorrência simultânea de quaisquer das condições anteriores.

De acordo com a cartilha divulgada do Censo realizado no ano de 2010 pelo Instituto Brasileiro de Geografia e Estatística - IBGE, no Brasil existem mais de 8,4 milhões de pessoas com deficiência visual, sendo aproxidamente 729 mil totalmente cegas e 7,7 milhões com baixa visão. A população residente no Brasil possui uma margem de 23,9% de pessoas com ao menos uma das deficiências investigadas, a saber, visual, auditiva, motora e mental ou intelectual o que corresponde a mais de 45 milhões de pessoas. A deficiência visual chama a atenção por possuir o maior índice investigativo dentre as outras, afetando 18,6% da população brasileira (CARTILHA..., 2015). A Figura 1 retirada da publicação *"Características Gerais da População, Religião e Pessoas com Deficiência"*, do IBGE remetem a estes índices:

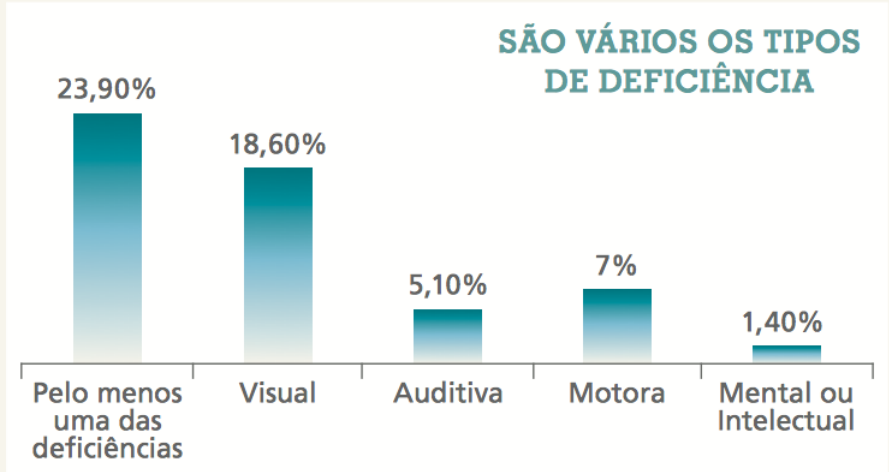

Figura 1: Proporção sobre a população total residente no Brasil no ano de 2010.

Quando se fala sobre a deficiência visual o tema mais discutido é sobre a inclusão social destas pessoas. Segundo (JESUS, 2007), para um cidadão, inclusão social significa ter desenvolvimento educacional, estar incluído nas atividades socioeconômicas de seu país, ter acesso às novas tecnologias da informação e conhecimento, para uma ação participativa junto à sua comunidade. Após a promulgação da Constituição de 1988, foi assegurado a igualdade de direitos no trabalho, proteção, integração social e educação social a todos os deficientes. O que esta constituição visa garantir é o reconhecimento social destas pessoas, a proteção e integração, pois elas são igualmente capazes de contribuir com a sociedade e de elaborar sua percepção social. (MATOS, 2003) afirma que: *"A discriminação, o desrespeito à diferença e a persistência de práticas segregativas e excludentes, refletem a falta de princípios éticos, morais e de cidadania."*. Baseando-se nos pontos levantados, incluir uma pessoa socialmente não se trata apenas de respeito, mas também em oferecer-lhes condições suscetíveis para o desenvolvimento informacional. A busca da igualdade, portanto, contribui no processo de aprendizagem dessas pessoas e não as trata como incapazes.

Na prática, os cegos ou pessoas com baixa visão não encontram dificuldades apenas ao caminhar na rua ou ao utilizar um transporte público, mas o preconceito, o uso da internet e o acesso a informação também são dificuldades que estas pessoas enfrentam no seu dia a dia. Este fatores influenciam na educação, na alfabetização, no aprendizado e integração social. Crianças brasileiras tem encontrado enormes dificuldade nas salas de aula o que evidencia que apenas a aceitação destas crianças nas escolas não é o suficiente. As escolas não estão preperadas estruturalmente para recebê-las e uma carência de professores capacitados para a formação educacional de crianças com deficiência visual é eminente. O mercado brasileiro de publicações também não oferecem suporte adequado. Este mercado foi avaliado em 2009, após quase 200 anos da criação do braille, e foi considerado incapaz de atender a demanda de livros transcritos o que desfavorece o acesso a informação básica, o processo de educação e o desenvolvimento cultural dos portadores desta deficiência (ALUNOS. . . , 2009). A tecnologia é capaz de oferecer este suporte por meios mais acessíveis proporcionando ao deficiente visual o acesso a informação e consequentemente influenciando positivamente na produtividade e no aprendizado destes (SILVA; TURATTO; MACHADO, 2003).

Fernando Botelho, aos 17 anos de idade, perdeu completamente a visão e encontrou enormes dificuldades para estudar. Segundo ele, a adaptação foi muito difícil mas ao passar a fazer uso da tecnologia como suporte ao ensino, seu desempenho aumentou consideravelmente tornando-se um dos melhores alunos da universidade. Ele conseguiu concluir a graduação, fez mestrado e hoje viaja o mundo onde sua carreira está sofrendo ascendência cada vez maior. No entanto, Fernando faz parte de um grupo muito pequeno de pessoas que tem acesso a este tipo de tecnologia. Em vista dos 180 milhões de pessoas no mundo que possuem deficiência visual e do alto custo para aquisição desta tecnologia, Fernando está desenvolvendo um programa de computador capaz de auxiliar pessoas com deficiência visual na educação. Segundo ele, é um direito de cada um participar da economia e do meio acadêmico (SOFTWARE. . . , 2012). Fundador do F123, projeto que a dois anos desenvolve um software de baixo custo que auxiliará as pessoas com deficiência visual nos seus estudos e profissionalmente, Fernando relata:

> O custo alto dos softwares, que já existem no mercado, e a falta de material apropriado para capacitação são os principais fatores para a exclusão de crianças e jovens cegos ou com baixa visão. Os softwares tipicamente usados para a leitura, ou ampliação de tela, custam o equivalente a dois ou três computadores. Este valor elevado impede governos, fundações e organizações de implementarem projetos na escala necessária para obter melhorias significativas, tanto na escolaridade quanto na empregabilidade de jovens com deficiência visual (SOFTWARE. . . , 2012).

O engenheiro e professor da Faculdade de Tecnologia de Curitiba, Frank Coelho, em uma entrevista cedida para um jornal televisivo de boa audiência, relatou sobre a experiência de vida de Fernando Botelho e o software que está desenvolvendo. Frank disse

que: *"Fernando é um gigante que anda no meio de nós mas ele não é um gigante que anda sozinho, ele anda nos ombros de outros gigantes."*. O Frank estava se referindo ao software livre onde a ideia defendida é que o conhecimento deve ser livre. O trabalho de Fernando foi feito com software livre. Frank defende que esta comunidade pode quebrar com paradigmas e revolucionar a civilização levando o conhecimento as pessoas. Este é o caminho para se conhecer a tecnologia e ajudar as pessoas. Ele também afirma que a tecnologia já está desenvolvida, mas ainda é preciso esforço e disposição para juntar e distribuir.

2.4.1 Contraste de Tela

Após algumas leituras notou-se que cores contrastantes é o conceito mais lembrado quando se fala de contraste de tela para pessoa com baixa visão. As cores contrastantes são cores que, quando usadas próximas umas das outras, produzem uma sensação de choque (GRIFFITH,). Na Figura 2, as cores mais contrastantes entre si estão diretamente oposta no círculo.

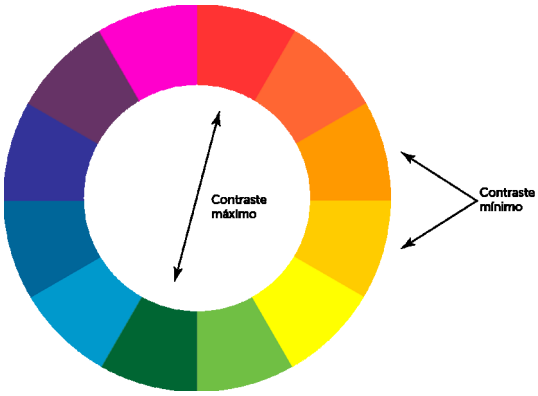

Figura 2: Matriz de contraste de cores (fonte).

Um artigo escrito por (KULPA; TEIXEIRA; SILVA, 2010) em sua pós-graduação em Design define um modelo de cores na usabilidade das interfaces computacionais para os deficientes de baixa visão. O artigo visa indicar boas práticas na criação ou adequação das interfaces utilizando cores que possibilitem ao deficiente de baixa visão navegar pelo ambiente virtual apenas com sua visão funcional, de forma confortável e satisfatória, sem o auxílio de tecnologias assistivas. Um dos modelos alcançam diversos tipos de diagnósticos de baixa visão e a está representado na Figura 3.

2.5 Formatos de Áudio Digital

O áudio digital refere-se a representação do som (captação por microfone) ou áudio analógico (representado por um vinil ou fita cassete) que é armazenado na forma de um

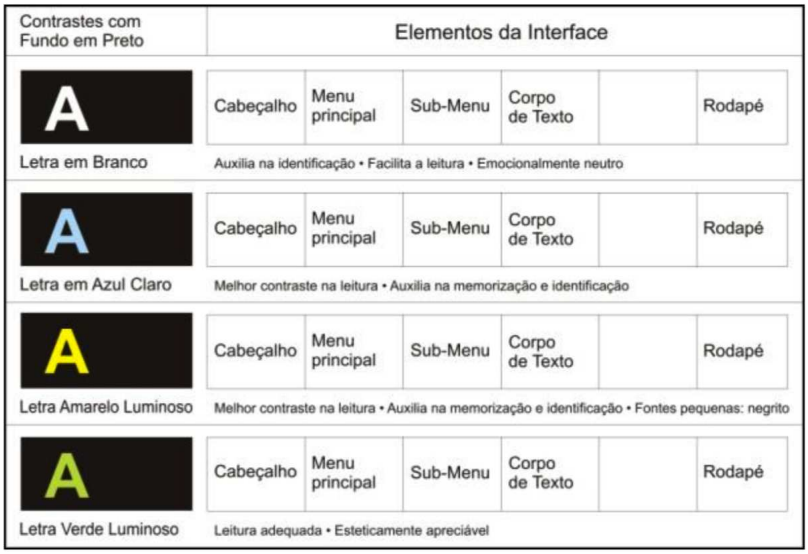

Figura 3: contrastes que alcançam diversos tipos de diagnósticos de baixa visão.

arquivo digital em uma unidade de computador ou outra mídia digital (como *Compact Disc*). O som é contínuo no tempo e para conseguir converter este sinal analógico em sinal de áudio digital, o conversor analógico-digital consiste em "tirar fotos" do sinal de áudio em intervalos constantes. Portanto, estas "fotos" são uma sequência de amostras que é representado por um sinal de tempo discreto originado da conversão de uma onda de som que, por sua vez, é um sinal contínuo. Assim, definimos **amostragem** como sendo a redução de um sinal contínuo em um sinal discreto. Em cada uma destas amostras é medido a intensidade do sinal que pode ser representada por 0s e 1s. Ou seja, um áudio digital é a representação binária de um som que é traduzida por um computador ou leitor de CD. Estes são capazes de reproduzir o som original.

Como vimos, o formato de áudio é uma forma de representar digitalmente um som ou um sinal de áudio. Algumas propriedades gerais de representação de som digital são:

***bitrate* ou taxa de bits:** que é o número de bits por segundos necessários para representar o sinal. O *bitrate* descreve, então, a velocidade com que o som é reproduzido (por exemplo, 320 000 bits por segundos ou 320 Kbps);

***number of channels* ou número de canais:** usa-se, tradicionalmente, um canal (som mono) ou dois canais (som estéreo);

***sampling frequency* ou frequência de amostragem:** define o número de amostras que são coletadas em um tempo pré determinado e a unidade de medição é dada em *Hertz*. As amostras são medidas em intervalos fixos.

Com o avanço e popularidade da digitalização do som, o áudio digital tem sido largamente usado. Um dos principais motivos é a facilidade no uso deste tipo de arquivo,

pois trabalhar com um sinal digital é muito mais simples se comparado ao sinal analógico. Os formatos são geralmente conhecidos por sua extensão e não pelo seu nome. Existem diversos formatos de áudio digital, mas primeiro vamos falar sobre o PCM.

2.5.1 Formatos não comprimidos

Os formatos não comprimidos possuem qualidade máxima pois não há alteração dos bits do ficheiro de áudio. Isto faz dos formatos não comprimidos útil para aplicações profissionais. O áudio digital é armazenado sem realizar a compressão dos dados e, portanto, demandam grande espaço em memória.

2.5.2 PCM

Com seu surgimento na década de 30 por Alec Reeves, o PCM (*do inglês, Pulse Code Modulation - Modulação por Código de Pulso*) é a forma mais antiga de digitalização do som e o mais utilizado devido à reprodução fiel do som, que é armazenado. Isso significa que os dados foram levados diretamente a partir da entrada, digitalizada e armazenada sem transformação. Não havia compressão de dados na época pois o poder de computação era escasso (WAGGENER, 1994).

Como explanado no início desta seção, a digitalização não é contínua, ou seja, não é constante como o som. Por possuir valores discretos (descontínuos), a digitalização sonora envolve dois parâmetros básicos: a frequência de amostragem e a profundidade de bit. A **profundidade de bit** (*bit depth*) trata da quantidade de bits de computador que são usados para representar cada amostra. Uma representação com um bit recebe apenas dois valores (0 e 1). Se tenho uma representação com 4 bits é possível receber 16 valores diferentes onde a amplitude de cada amostra é um dos 16 valores possíveis (WAGGENER, 1994). A taxa de amostragem e a amplitude influenciam diretamente na qualidade do áudio. Sony e Philips desenvolveram a tecnologia PCM e criaram o *Compact Disc*. O CD possui 44100 amostras por segundo e uma amplitude de 16 bits o que torna a qualidade alta.

2.5.3 Formato WAV

O formato *Waveform Audio Format* ou WAVE (os ficheiros deste formato utilizam a extensão *.wav) possui o seu áudio codificado em PCM. Criado pela Microsoft, este formato de áudio digital é nativo do sistema operativo Windows e é compatível com quase todos os tocadores atuais. Um ponto negativo é o tamanho de seu arquivo que chega a possuir uma média de 10 MB por minuto (SERRA, 2002).

O WAV é derivado do padrão IFF criado pela *Eletronic Arts Interchange File Format* onde seu conteúdo é divido em blocos denominados *chunks*. Existem diversos

tipos de *chunks* em um arquivo WAV, mas apenas dois são obrigatórios (além do chunk *header* que especifica o formato do áudio).

fmt chunk : define algumas informações sobre o formato do arquivo tais como número de canais, sua taxa de amostragem e o tamanho de seus *samples*;

data *chunk* : responsável por armazenar a sequência de *samples*.

A Figura 4 ilustra o que foi explicado sobre a estrutura geral de um formato WAVE.

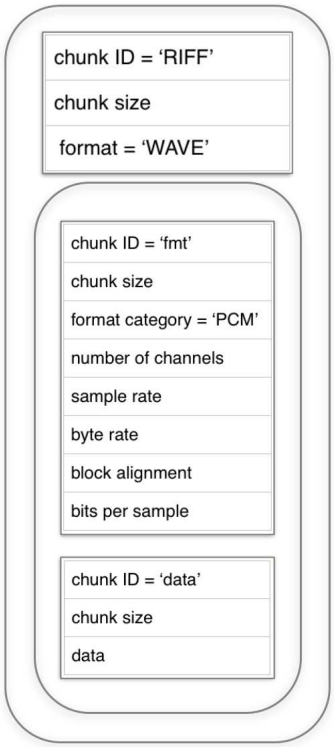

Figura 4: Diagrama Geral do formato WAVE.

2.5.4 Formato AIFF

O formato *Audio Interchange File Format* (a extensão destes ficheiros pode ser *.aiff ou *.aif) é bastante similar ao WAV. Foi criado pela Apple e portanto é bastante popular e nativo em seu sistema operacional. O AIFF possui áudio em formato PCM porém não é tão difundido quanto o formato WAV. Como o nome sugere, o AIFF também usa o método IFF para armazenar seus dados e também possui dois *chunks* obrigatórios: o *common chunk* e o *sound data chunk* (MURATNKONAR, 2014). A Figura 5 ilustra, de forma geral, como um formato AIFF de áudio é estruturado.

***common chunk*:** define algumas informações fundamentais do formato do como visto na Figura 5;

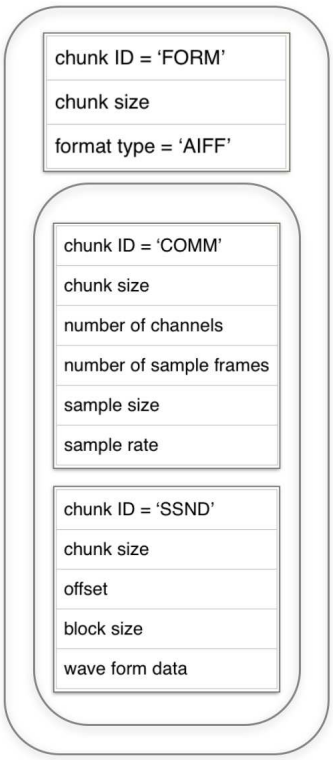

Figura 5: Diagrama Geral do formato AIFF.

sound data *chunk*: responsável por armazenar a sequência de *sample frames*.

2.5.5 Formatos comprimidos

Os formatos comprimidos possuem qualidade inferior aos formatos não comprimidos. Todavia, dependendo da compressão feita, mesmo com um pouco da perda da qualidade, a alteração é imperceptível. Por conta da compressão é possível obter um ficheiro de áudio com qualidade boa e com o espaço ocupado em memória reduzido.

2.5.6 Formato MP3

O MPEG-1 Layer 3 (popularmente conhecido com MP3) é a primeira versão do codec MPEG que é utilizada, atualmente, para codificar áudio. O MPEG, do inglês *Moving Picture Experts Group* é um padrão que define técnicas de compressão de áudio e vídeo. A Camadas (*Layers*) diferem-se em termos de qualidade, sendo as primeiras dotadas de qualidades superiores e consequentemente voltadas para uso profissional de áudio como em estúdios de gravação. A Camada 3 foi adotada para o consumidor final por ser capaz de gerar arquivos com boa taxa de compressão e áudio de boa qualidade. O MP3 é construído a partir de pequenas partes chamadas *frames*. Cada frame, por sua vez, é composto por um *header* e um bloco de dados (ERROR, 1999). A Figura 6 melhor define esta estrutura.

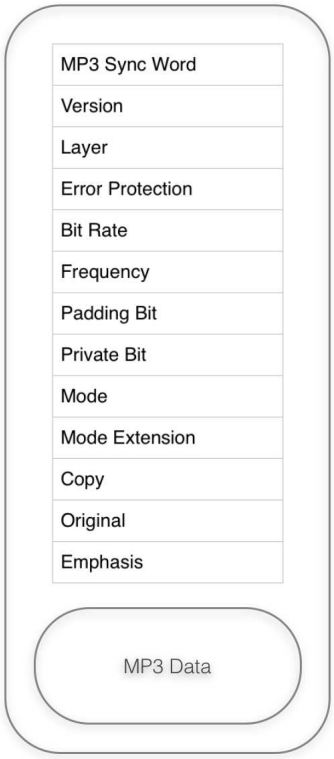

Figura 6: Diagrama Geral representando um *frame* do formato MP3.

O MP3 se popularizou através de um software criado por Shawn Fanning chamado Napster. Apesar da rede Napster ter tido uma vida curta (cerca de 2 anos), o programa fez bastante sucesso sendo o percursor do compartilhamento de arquivos de áudio em formato MP3. Por conta da dificuldade em encontrar áudios em formato MP3, Fanning criou o programa para compartilhar estes ficheiros de áudio entre amigos. Usando da tecnologia P2P (*peer to peer*) e pela popularidade entre os amigos, o programa foi copiado para diversas outras pessoas alcançando pessoas ao redor do mundo (ALECRIM, 2006). Dentre os formatos de áudio, o MP3 é o mais popular. O grande problema dele é sua patente.

2.5.7 Formato Ogg

O resultado de um encapsulamento Ogg é chamado de ***Pyshical Ogg Bitstream***. O Ogg encapsula dados comprimidos criado por codecs. Os dados comprimidos são fluxos de bits de mídia que é chamado de ***logical bitstreams*** e pode ser representados por um fluxo simples de áudio Vorbis, múltiplos fluxos de áudio ou fluxos de áudio e vídeo multiplexados. Um *logical bitstream* é estruturado, ou seja, ele é dividido em uma sequência de ***packets***. Os pacotes são criados pelo codificador de *logical bitstream* e estes pacotes apenas tem representação para o codificador: “Please note that the term packet is not used in this document to signify entities for transport over a network” (RFC_3533, 2003).

O Ogg oferece suporte para o transporte do fluxo de bits lógicos como enqua-

dramento e intercalação para diferentes fluxos. O formato também é capaz de detectar corrupção e recuperar-se após algum erro de análise. Marcos de posição é suportado e, dessa forma, torna-se possível o acesso aleatório direto de posições arbitrárias na *bistream*. Outra forte característica é a capacidade de streaming, onde não é necessário a construção completa do *bitstream* para a transmissão do mesmo. Outros suporte, e não menos importante, é a pequena sobrecarga referente a largura de banda do bitstream.

O *Pyshical Ogg Bitstream* consiste de vários *logical bitstreams* intercalados em ***Pages***. O *logical bitstream* são indentificados por um número de série único no cabeçalho de cada *page*. As *pages* são intercaladas simultaneamente e não precisam seguir uma ordem regular, mas precisam ser consecutivos dentro de um *logical bitstream*. A desmultiplexação Ogg é capaz de reconstruir o *logical bitstream* original. Cada *page* contém apenas um tipo de dado, uma vez que pertence a um único *logical bitstream*. As *pages* possuem tamanhos variáveis e tem um cabeçalho contendo encapsulamento e recuperação de erros de informação. Cada *logical bitstream* em um *physical Ogg bitstream* começa com uma páginal inicial especial **bos** (*beginning of stream*) e termina com uma página especial **eos** (*end of stream*) (RFC_3533, 2003).

Uma página bos contém informações para identificar o tipo de codec e pode conter informações para configurar o processo de decodificação. Ela também deve conter informações sobre os meios de comunicação codificada. Para exemplificar, uma página bos, para um áudio, deve conter a taxa de amostragem e o número de canais. Por convenção, os primeiros bytes de uma página bos contém informações necessárias para identificação do *codec*. O Ogg também permite mas não obriga pacotes de cabeçalhos (*header packets*) secundários após a página bos para o *logical bitstream* e estes devem preceder quaisquer pacotes de dados (*data packtes*) em qualquer *logical bitstream*. Este pacotes de cabeçalhos são enquadrados em um número integral de páginas e estas, por sua vez, não contém quaisquer pacotes de dados. Portanto, conclui-se que um *physcial Ogg bitstream* começa com as páginas de todos os *logical bitstreams* contendo um pacote de cabeçalho inicial por página, seguido por pacotes de cabeçalhos subsidiários de todos os fluxos e, por fim, seguido pos páginas contendo pacotes de dados (RFC_3533, 2003).

A especificação de encapsulação de um ou mais *logical bitstreams* é chamado de ***media mapping*** ou mapeamento de mídia. A (RFC_3533, 2003) exemplifica um mapeamento de mídia citando o "Ogg Vorbis", que usa a estrutura Ogg Vorbis para encapsular os dados de áudio codificados para armazenamento baseado em arquivo e transmissão baseada em fluxo. O Ogg Vorbis fornece o nome do *codec* e algumas informações referentes ao áudio codificado. Ele também possui duas páginas adicionais de cabeçalho. A página bos do Ogg Vorbis começa com o byte 0x01, seguido por "Vorbis" totalizando 7 bytes de identificador. O *codec* Vorbis será visto na seção 2.4.2.

2.5.7.1 Mutiplexagem

O formato possui dois tipos de multiplexagem: *grouping* (agrupamento) e *chaining* (encadeamento). No ***grouping*** a multiplexação é simultânea onde vários *logical bitstreams* são intercalados em um mesmo *physical bitstream* como, por exemplo, a intercalação de um fluxo de vídeo com diversas faixas de áudio. É importante ressaltar que na multiplexagem *gouping* todas as páginas bos de todos os *logical bitstreams* devem aparecer juntas no início do *physical Ogg bitstream*. Por outro lado, as páginas eos não precisam estar lado a lado. A multiplexagem ***chaining***, por sua vez, foi definido para proporcionar um mecanismo simples de concatenação do *physical Ogg Vorbis* onde os *logical bitstreams* completos são concatenados. O *chaining* são frequentemente usados em aplicações de transmissão. Não há sobreposição dos *bitstreams* pois a página eos de um dado *logical bitstream* é imediatamente seguido pela página do próximo bos (RFC_3533, 2003). Os dois tipos de multiplexagem podem ser usados em conjunto. A Figura 7 é um diagrama que representa os dois tipos de multiplexagem.

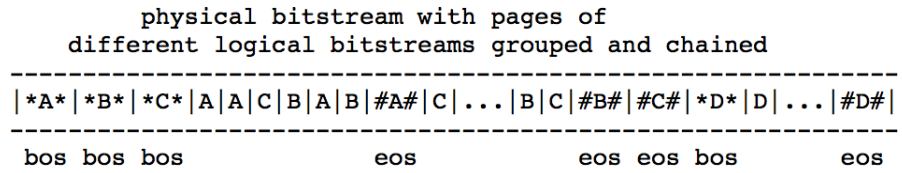

Figura 7: Diagrama exemplificando as multiplexagens (RFC_3533, 2003)

O Ogg não sabe nada sobre os dados do *codec* que ele encapsula e é, portanto, independente de qualquer codec de mídia. A exceção para aquilo que o Ogg conhece é que cada *logical bitstream* pertence a um *codec* diferente, os dados do *codec* vem em ordem e possuem marcadores de posição (são *granule positions* que serão abordados mais a frente). O recipiente Ogg não possui conceito de tempo. No entanto, o Ogg sabe sobre o aumento sequencial dos marcadores de posição. Isto faz do Ogg uma estrutura genérica mas que realiza o encapsulamento de fluxos de bits de tempo contínuo.

O processo de multiplexação ocorre no nível de página. Porém, os *codecs* fornecem fluxos de bits que são entregues ao Ogg em pacotes de tamanho arbitrário e não limitado. As páginas, em contra partida, possuem tamanho máximo de 64 Kbytes. Desta forma, os pacotes, na maiora das vezes, precisam ser distribuídos ao longo de várias páginas. Para realizar essa distribuição e facilitar o processo, o Ogg quebra um pacote em **segmentos** de 255 bytes sendo o último segmento mais curto. Os segmentos são apenas uma construção lógica e não tem um cabeçalho para si. Uma *flag* no cabeçalho da página informa se uma página contém informações de um pacote de continuação da página anterior (RFC_3533, 2003). Para um melhor entendimento, a Figura 8 é um diagrama que exemplifica a relação entre páginas, pacotes e seguimentos.

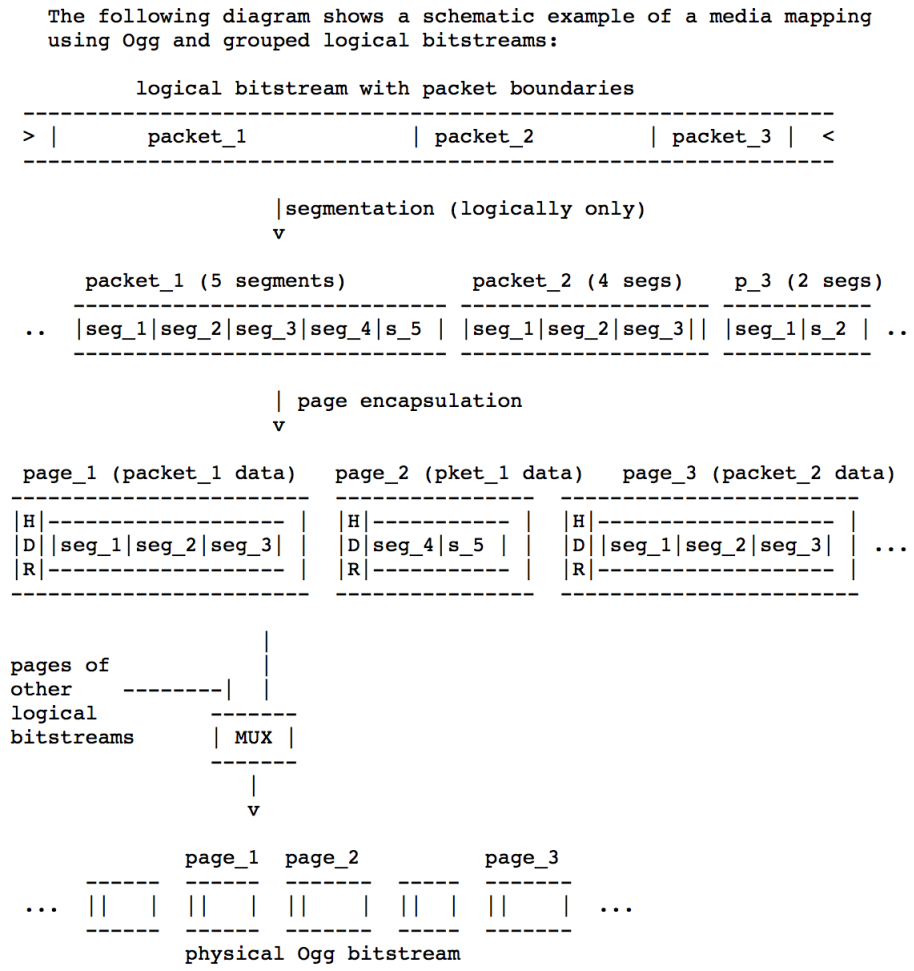

Figura 8: Representação da disposição dos segmentos (RFC_3533, 2003)

Um cabeçalho de página contém todas as informações necessárias para a desmultiplexação dos fluxos de bits lógicos e para executar a recuperação de erros básicos e marcos para a procura. Cada página é uma entidade auto-suficiente de modo que o mecanismo de página de decodificação pode reconhecer, verificar e lidar com páginas simples em um momento sem ser necessário todo o fluxo de bits. A Figura 9 define o formato para um cabeçalho de página.

Segundo (RFC_3533, 2003), os campos da estrutura da Figura 9 possuem os seguintes significados:

- ***capture_pattern***: cada página começa com uma *string* de quatro bytes "OggS". As letras "O" e "S" devem ser maiúsculas e suas representações na tabela ASCII dadas em hexadecimal são 0x4f e 0x53, respectivamente. A letra minúscula "g" é representada por 0x67. Se a sincronização for perdida, este campo ajuda o decodificador na recuperação da sincronização;

- ***version***: especifica a versão do arquivo Ogg e possui 1 byte para sua representação;

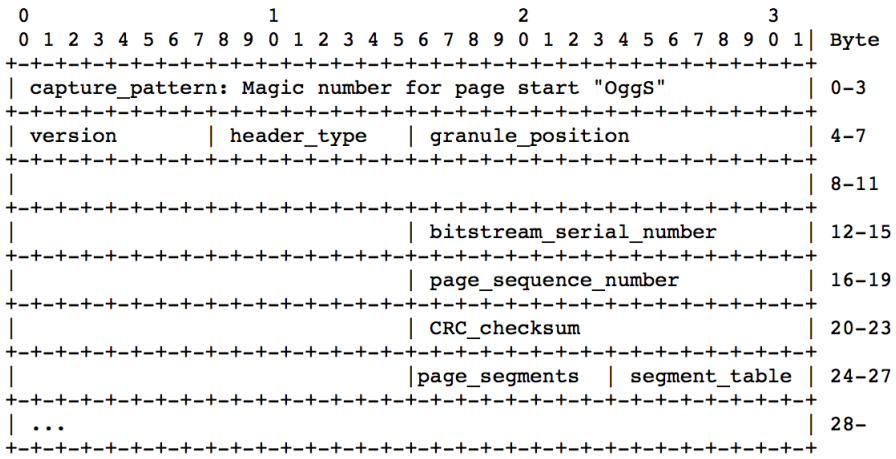

Figura 9: Formato de cabeçalho de uma página (RFC_3533, 2003)

- ***header_type***: Este campo possui apenas um byte. No entanto, os bits identificam o tipo específico de cada página:

  bit 0x01: se este bit for igual a 1, significa dizer que a página contém dados de um pacote de continuação da página anterior. Caso seja 0, a página está iniciando um novo pacote;

  bit 0x02: se o bit estiver setado como um, então é uma página bos. Se o bit for 0, significa que não é uma primeira página;

  bit 0x04: se o bit estiver setado como um, então é uma página eos. Se o bit for 0, significa que não é a última página.

- ***granule_position***: Os 8 bytes (ou 64 bits) que representa este campo, contém informações referente a posição. O Ogg não contém informações de tempo, porém o tempo é mapeado referente a posição de um dado digital em uma *stream*. O significado real deste campo em relação ao tempo depende de cada *codec*;

- ***serial_number***: o *logical bitstream* é identificado com um único número de série neste campo de 4 bytes;

- ***sequence_number***: se o campo anterior identifica um *logical bitstream* este campo contém o número de sequência da página que é incremetnado separadamente. O campo também possui 4 bytes e através dele o decodificador consegue identificar a perda de página;

- ***CRC_checksum***: contém CRC checksum de 32 bits cujo polinômio gerador é 0x04c11db7;

- ***page_segment***: este campo possui um 1 byte de tamanho e fornece o número de entradas de segmentos codificados na tabela de segmentos;

- ***segment_table***: contém informações de *lacing values* de todos os seguimentos contido na página. Cada byte deste campo contém um valor de laço e, portanto, o número de bytes deste campo é igual a quantidade de segmentos existentes na página.

2.5.8 *Codec* Vorbis

O *codec* Vorbis codifica blocos PCM de curta duração em pacotes de bits de dados comprimidos. Ele não fornece nenhum tipo próprio de enquadramento, sincronização ou proteção contra erros. Para isso, Vorbis utiliza do formato de fluxos de dados do Ogg para disponibilizar sincronização, recaptura de sincronização após erros, marcadores durante a procura e informação suficiente para separar corretamente os pacotes de dados codificados para obter o pacote original. Segundo sua especificação (XIPH.ORG, 2012), Vorbis é apenas um método de aceitação de entrada de áudio, dividindo este áudio em *frames* (quadros) individuais comprimidos que são pacotes não formatados. Basicamente, o decodificador aceita estes pacotes em sequência, decodifica-os e os sintetiza no fluxo do áudio original.

Para obter arquivos cada vez menores e com uma qualidade constante, o Vorbis utiliza uma codificação de débito binário variável. O **VBR** (*Variable Bit Rate*) é um algoritmo que define qual a melhor taxa de bit para cada frame da música variando, assim, a quantidade de transferência de bits por segundo mas mantendo a qualidade constante (BECKER..., 2014). Ou seja, O Vorbis é um formato de taxa variável e seus pacotes não possuem tamanho fixo e depende do parâmetro de qualidade. O nível de qualidade varia de 0 a 10 com mudanças feitas de 1 em 1. A qualidade 0 corresponde a 64 Kbps, 5 por volta de 160 Kbps e 10 cerca de 400 Kbps. O nível cinco de qualidade já é suficiente para se aproximar de uma qualidade de áudio de CD. Portanto, ao usar o VBR, o Vorbis consegue atingir qualidade de som semelhante ao original e com melhor compressão. A taxa de bits varia de 16 Kbps a 500 Kbps por canal. O número de canais de áudio independentes suportados podem chegar a 255 (XIPH.ORG, 2012). No entanto, o nível 3 é mais utilizado por conseguir, a partir de 110 Kbps, gerar ficheiros de áudio menores e qualidade superior a ficheiros MP3 com 128 Kbps.

2.5.8.1 Algortimo de compressão

Os métodos usados para a compressão de áudio geralmente pertencem a duas categorias: métodos que trabalham no domínio do tempo e métodos que trabalham no domínio da frequência. Todos os formatos conhecidos que trabalham no domínio da frequência usa a transformada discreta do cosseno modificada. O **MDCT** (*Modified Discrete Cosine Transform*) realiza a tranformação para o domínio da frequência a partir do domínio do tempo. O ***window shape*** que contém amostras PCM com duas variações de com-

primentos especificados em 2048 ou 512 amostras. O Vorbis usa o MDCT para aplicar transformação com janelas sobrepostas (XIPH.ORG, 2012).

2.5.8.2 Estruturação

O Vorbis é estruturado em pacotes e utiliza quatro tipos diferentes. O três primeiros tipos marcam três tipos diferentes de cabeçalho e devem estar dispostos na seguinte ordem: cabeçalho de identificação, cabeçalho de comentário e cabeçalho de configuração. Após os três pacotes de cabeçalho, todos os pacotes subsequentes são pacotes de áudio. A estrutura, de uma forma geral, é ilustrada na Figura 10.

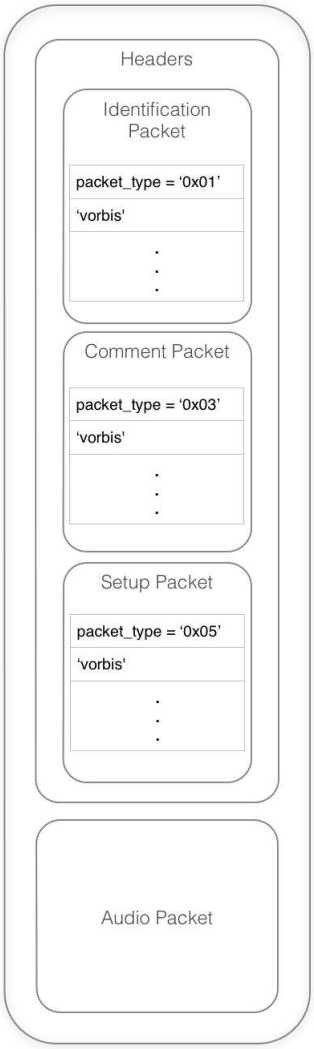

Figura 10: Diagrama Geral representando uma estrutura Vorbis.

Todos os pacotes cabeçalhos começam com um byte para identificação do tipo de cabeçalho e mais seis bytes para identificação da *string* "vorbis". O campos seguintes são específicos de cada pacote. O ***identification packet*** identifica o fluxo de bits como Vorbis e contém informações sobre a versão do Vorbis usado, o número de canais de áudio e a taxa de amostragem. O segundo pacote é o ***comment header*** e contém

metadados que são comentários de usuários referentes ao pacote de áudio tais como nome da música, nome dos artistas ou o nome da gravadora. Este campo não é necessário para a decodificação do áudio, mas é um pacote obrigatório e deve ser identificado e ignorado adequamente para não corromper o arquivo. O Vorbis sugere alguns nomes de campos mas eles não são obrigatórios e outros também podem ser usados. A lista contém as seguintes sugestões: TITLE, VERSION, ALBUM, TRACKNUMBER, ARTIST, PERFORMER, COPYRIGHT, LICENSE, ORGANIZATION, DESCRIPTION, GENRE, DATE, LOCATION, CONTACT e ISRC. A maioria das informações necessárias para inicializar o decodificador estão contigas no ***setup header***. Ele contém informações de configuração do *codec* e *codebooks* contendo o VQ completo e o código de Huffman. Por último, temos o ***audio packet*** que contém os dados de áudio do arquivo (XIPH.ORG, 2012).

No processo de decodificação, a primeira etapa da decodificação de pacotes de áudio é para ler e verificar o tipo de pacote. Um pacote não-audio quando um pacote de áudio é esperado indica corrupção de fluxo. O decodificador deve ignorar o pacote e não tentar decodificá-lo para áudio. Os pacotes de áudio começam com um único bit que precisa sempre ser 0 (XIPH.ORG, 2012).

2.5.9 Licença de uso

É intenção dos desenvolvedores Ogg Vorbis que o formato seja utilizável sem preocupações de propriedade intelectual e, por isso, são de domínio público. *"Ogg Vorbis is a fully open, non-proprietary, patent-and-royalty-free (...)"* (XIPH.ORG, 2000). O Ogg Vorbis está sob licença BSD e os termos de licença estão descritos e acessíveis em (XIPH.ORG, 1994).

2.6 Reuso de Requisitos

Segundo (SOMMERVILLE, 2004), "quando se desenvolve requisitos para um novo sistema, deve-se, na medida do possível, reutilizar os requisitos de outros sistemas que foram desenvolvidos para a mesma área da aplicação". Para (RENAULT, 2009), o reuso de requisitos é "a tendência de se utilizar o conhecimento adquirido em experiências anteriores no levantamento de requisitos e impulsionar a reutilização desse conhecimento".

Por meio do método analítico, o reuso de requisitos é uma das técnicas conhecidas da atividade de elicitação de requisitos que visa analisar e estudar a viabilidade de reutilização de especificações existentes de sistemas legados ou sistema com funcionalidades de negócio similares. As principais vantagens do uso desta técnica são: melhorar a qualidade e elicitação de requisitos e reduzir o custo e tempo envolvidos no desenvolvimento do software visto que os requisitos reutilizados já foram analisados e validados em outros

sistemas. Requisitos reutilizados são mais facilmente assimilados por usuários pois, de certa forma, já são conhecidos por eles (REFLECTZ, 2011).

2.7 User Stories

As *User Stories* ou Histórias de Usuário apresentam algumas das características dos casos de usos ou das declarações tradicionais de requisitos e tem sido utilizada largamente em ambiente ágil (COHN, 2004). No processo de levantamento de requisitos, a *User Story* é usada como instrumento de escrita que descreve a especificação de uma funcionalidade do software em um cartão (BECK; FOWLER, 2001). Um formato simples de *user story* é sugerida por (COHN, 2004) e apresenta a seguite sintaxe: ***Como um*** *tipo de usuário* ***eu quero*** *capacidade ou funcionalidade* ***de modo que*** *o valor do negócio ou benefício.* A Figura 11 mostra um exemplo de cartão de *user story.*

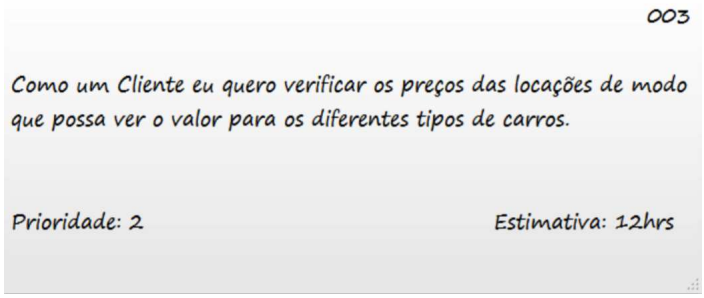

Figura 11: Um cartão de *User Story.*

Quando uma *user story* é implementada, um teste de aceitação mais formal deverá ser escrito para assegurar que os objetivos da *user story* sejam cumpridos. Os testes de aceitação são criados a partir de critérios de aceitação que são descritos no verso do cartão da *user story.* Os critérios de aceitação define os limites de uma *user story* e são utilizados para confirmar implementação da *user story* e validar o seu correto funcionamento conforme o esperado (COHN, 2004). A Figura 12 mostra um exemplo de critério de aceitação.

(GRENNING, 2011) observou que os textos em uma *User Story* que descrevem os critérios de aceitação possuem os mesmo dados que um caso de uso. Mediante isso, o autor sugere que a história do usuário corresponde ao cenário do caso de uso principal, e que os testes de aceitação da história correspondem aos fluxos alternativos do caso de uso. Dando continuidade a explanação, (GRENNING, 2011) diz que os critérios de aceitação estão associados às histórias de usuário para especificar os requisitos de software que atendam as regras de negócio.

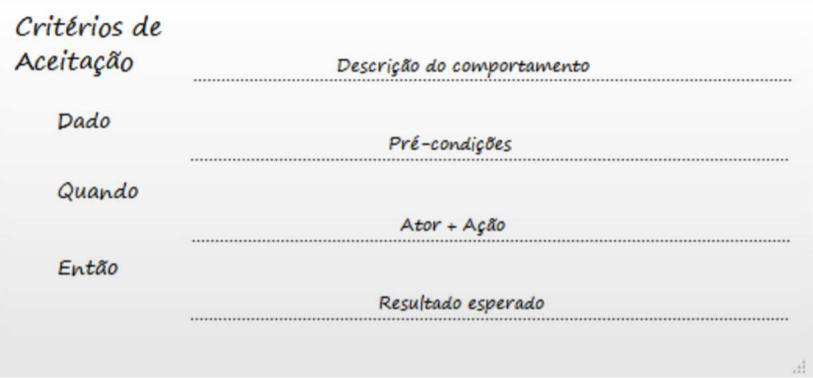

Figura 12: Um cartão de Critérios de Aceitação.

3 Metodologia

3.1 Levantamento Bibliográfico

A pesquisa bibliográfica foi feita, basicamente, por assunto, por autor e por título. A pesquisa realizada por assunto foi a mais utilizada e termos adequados foram usados para se obter um pesquisa mais efetiva para o entendimento e desenvolvimento do trabalho tais como *audiobooks*, não-videntes, deficientes visuais, educação, ogg vorbis, livorbisfile, libvorbis, libvorbisenc, compressão de áudio, formatos de áudio, WAV, AIFF, MP3, PCM, entre outros. Também foram feitas pesquisas por assunto a respeito de ferramentas necessárias para o desenvolvimento do trabalho tais como fórmulas latext, player ogg, sox play, dump ogg, bibtex models example, Qt Creator e Qt Designer, entre outros. Para a pesquisa feita por autor e título é necessário que já se saiba qual autor ou obra são relevantes para o tema escolhido como, por exemplo, a pesquisa pelo autor Tanenbaum. Levantamento por assunto foi bastante utilizado em pesquisa na internet usando catálogos e mecanismos de busca em sites como o Google Acadêmico (GOOGLE, 2014), Google com pesquisa *web* e com filtro para livros, *ACM Digital Library* (ACM, 2014) e Scielo (SCIELO, 2014). Ideias e dicas dadas pelo prof. Dr. Edson Júnior foram de suma importância principalmente para uma determinação de "um ponto de partida".

Através do levantamento bibliográfica foi possível listar e consolidar citações de trabalhos fundamentais para o tema ou algo similar ao que foi proposto neste trabalho.

3.2 Ferramentas utilizadas

As ferramentas utilizadas para o desenvolvimento do trabalho são descritas, em poucas palavras, para qual propósito cada uma foi usada e qual a versão utilizada.

Por fornecer ferramentas nativas que ofereceram grande suporte para o desenvolvimento deste trabalho, pelo formato Ogg Vorbis fornecer API de fácil instalação e uso em distribuições Unix e pelo conhecimento prévio dos sistemas operacionais o Ubuntu 14.04 LTS foi adotado para um primeiro momento e o OS X Yosemite versão 10.10.4 para um segundo momento para o ambiente de desenvolvimento. O uso dos dois sistemas não influenciou no desenvolvimento, ambos são distribuição Unix sendo pessoal o fator motivacional. O compilador utilizado foi o gcc versão 4.8.2.

O editor de texto utilizado para o desenvolvimento dos códigos fonte em linguagem de programação C foi o Sublime Text 2. Ele foi escolhido por possuir uma interface limpa, por fornecer uma prévia visualização de todo o documento e, principalmente, pelos

comandos que ele oferece para facilitar a codificação. Podemos citar como exemplo a troca de linhas sendo possível mover uma linha ou um bloco inteiro de código, cursores múltiplos possibilitando a escrita em diversas parte do código simultaneamente, busca por palavras de mesmo nome e a busca e substituição de palavras específicas com uma ou múltiplas seleções.

O Latex foi utilizado para desenvolver o trabalho escrito e escolhido por gerar, como saída, um pdf com alta qualidade tipográfica e totalmente formatado. A versão utilizada foi pdfTeX versão 3.1415926-2.5-1.40.14.

Inicialmente foi utilizada a ferramenta hexdump nativa do sistema operacional Unix para mostrar os dados binários em hexadecimal com o intenção de facilitar o entedimento do formato de áudio e validar os dados inseridos posteriormente. Entretanto, a feramenta hexdump foi substituída pela oggz versão 1.1.1 por apresentar de forma mais limpa e segmentada os pacotes do formato Ogg e por conter informações adicionais como: número do pacote, tamanho do pacote, sequência, dentre outras.

A versão v.14.4.1 da ferramenta sox foi utilizada para executar os arquivos de áudio *.ogg com o intuito de verificar se o arquivo não foi corrompido devido as constantes modificações do formato para inserção do pacote de marcação de conteúdo e da inserção dos metadados no cabeçalho de comentários.

A biblioteca libvorvis foi utilizada como suporte ao uso da compressão de dados e integrada no Editor desenvolvido. A rotina é capaz de comprimir os dados contidos no formato e sua posterior recuperação.

O Qt Designer 5.4.2 foi utilizado para a construção da interface gráfica da aplicação que executa o formato especificado. O Tocador foi desenvolvido em C++ e, para execução do áudio, a rotina do framework SDL (Simple Directmedia Layer) versão 1.2.15 foi configurada e integrada na aplicação. O QtSpeech foi utilizado como suporte ao uso de sintetizador de voz no Tocador e é capaz de utilizar a voz sintetizada do próprio sistema operacional.

O Qt é um framework multiplataforma que possui deversos módulos para desenvolvimento de software. Como alguns módulos do Qt foram utilizados para a construção do Tocador e de sua interface gráfica, tais como QtCore, QtGui, QtWidgets e QtDesigner, para a cobertura de testes, o módulo QtTest foi selecionado para a implementação dos testes unitários.

Para o controle de versão, o GitHub foi a ferramenta utilizada. O projeto está acessível em BryanFernandes/player_rab-ogg.

Dada a complexidade para a montagem do ambiente de desenvolvimento viu-se a necessidade de automatiza-lo. Dessa forma, o próprio sistema operacional será o responsável por construir um ambiente capaz de compilar, executar e testar o Tocador. Com

suporte aos sistemas operacionais OS X e Linux, o Vagrant foi a ferramenta escolhida para este fim. A partir de um script, ele é capaz de criar e recriar ambientes em qualquer lugar de forma simples e descomplicada. Para isso, o Vagrant faz uso das soluções de virtualização mais comuns como o Virtualbox que será utilizado neste trabalho.

A ferramenta CodeClimate foi utilizada para garantir uma melhor qualidade de código e prevenção de potenciais *bugs*. Esta ferramenta realiza a análise estática do código fonte e notifica a equipe de desenvolvimento quando potenciais bugs podem ocorrer no projeto. A ferramenta também é capaz de manter um registro das *issues* encontradas.

Por apresentar fácil utilização e integração com outros sistemas, o Travis CI foi a ferramenta escolhida para automação do processo e engenharia do produto. A ferramenta integra-se ao GitHub indicando quando em cada commit ocorreu sucesso ou falha no processo de automatção. O Travis também mantém um histórico com logs das execuções realizadas além de notificar por e-mail a equipe de desenvolvimento ao final de cada execução realizada.

3.3 Proposta anterior

A proposta de solução foi baseada no trabalho desempenhado por (REIS, 2013). Como ponto inicial ao desenvolvimento da solução tecnológica, a primeira coisa a ser feita foi entender a proposta de (REIS, 2013) e, para isso, foi feito um estudo em cima de sua monografia. Conforme documentado, o sistema possuem três componentes, sendo eles: o Tocador, o Editor RAB e o Core como mostra a Figura 13.

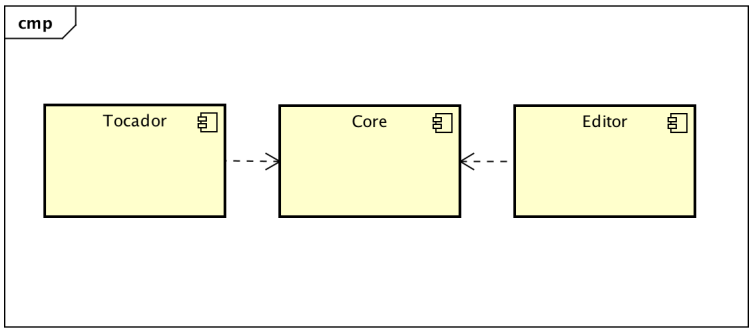

Figura 13: Diagrama de Componentes.

O Core é a biliboteca estática comum e importante ao funcionamento do Tocador e do Editor. A Figura 14 apresenta a biblioteca estática de forma mais detalhada.

O diagrama de classes do Tocador está representado na Figura 15. O componente foi separado nas camadas Modelo, Visão e Controladora segundo o padrão MVC. O pacote core representa a biblioteca estática.

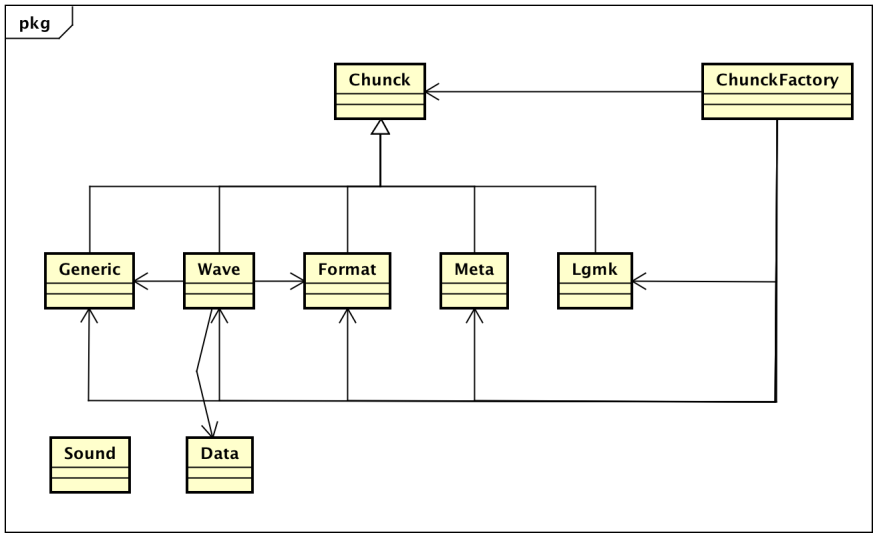

Figura 14: Diagrama de Classe do Core.

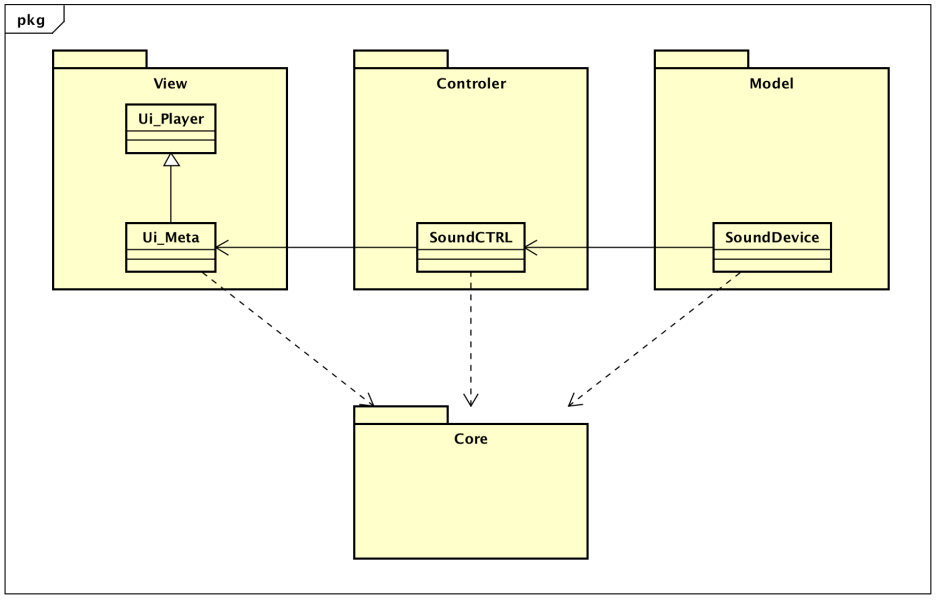

Figura 15: Diagrama de Classe do Tocador.

E, por fim, a Figura 16 mostra o diagrama de classe do Editor. A classe *Main* utiliza os serviços das classes da biblioteca Core e implementa as funções de iteração com usuário através de linha de comando.

A proposta de melhoria para o trabalho anterior considerou os problemas levantados e traçou, como objetivo, uma proposta inovadora sobre os aspectos sociais, econômicos, tecnológico e, principalmente, sobre a inclusão de pessoas que possuam alguma limitação para uso do sistema. Portanto, a solução trata da inclusão social dos deficientes visuais oferecendo a eles suporte ao ensino com baixo custo do uso da tecnologia.

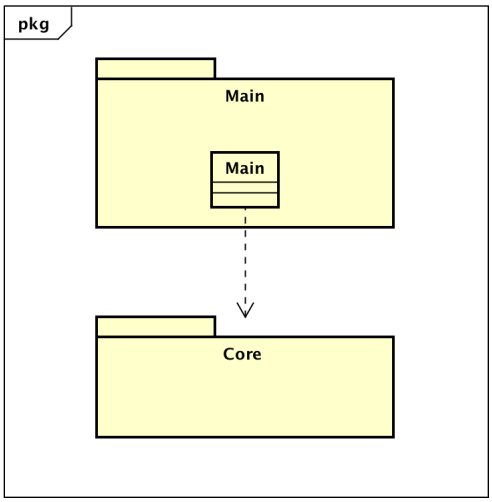

Figura 16: Diagrama de Classe do Editor RAB.

3.4 Entendendo o Ogg Vorbis

O segundo passo foi o estudo de um novo formato: Ogg Vorbis desenvolvido pela fundação Xiph.org. Foi feito um estudo minucioso em cima do documento de especificação do formato Ogg Vorbis para conhecer a sua estrutura e o tipo de suporte que ele oferece. A ferramenta *hexdump* utilizada por (REIS, 2013) foi usada como suporte para um melhor entendimento do formato *.ogg. A Figura 17 mostra o resultado da execução do hexdump em um arquivo *.ogg cujo comando é: $ hexdump -C <file_name>.

No entanto, os dados ainda ficaram muito confusos e de difícil interpretação. Após uma pesquisa verificou-se a existência de uma outra ferramenta também nativa no sistema operacional Ubuntu e voltada para os arquivos com extensão *.ogg e possui a mesma finalidade da ferramenta hexdump. A diferença entre elas está na forma com que os dados são apresentados. Podemos verificar na Figura 18 como o oggz-dump estrutura os dados. O comando para execução da ferramenta é dado no terminal e possui o seguinte formato: $ oggz-dump <file_name>.

É notável a diferença e a facilidade com que a ferramenta oggz informa sobre os pacotes contidos em um arquivo *.ogg. Após o comando é possível verificar os pacotes separadamente bem como sua informações tais como número do pacote, informações de grânulo, o "tempo" em que aquele pacote é lido, o seu tamanho, entre outras informações. Além das informações do pacote também é possível visualizar seu conteúdo e logo percebemos que os pacotes de número 0, 1 e 2 são os pacotes cabeçalhos. A identificação dos pacotes cabeçalhos pode ser percebida pois após o primeiro byte do pacote, os 6 bytes subsequentes contém a string "vorbis" onde cada byte representa uma letra.

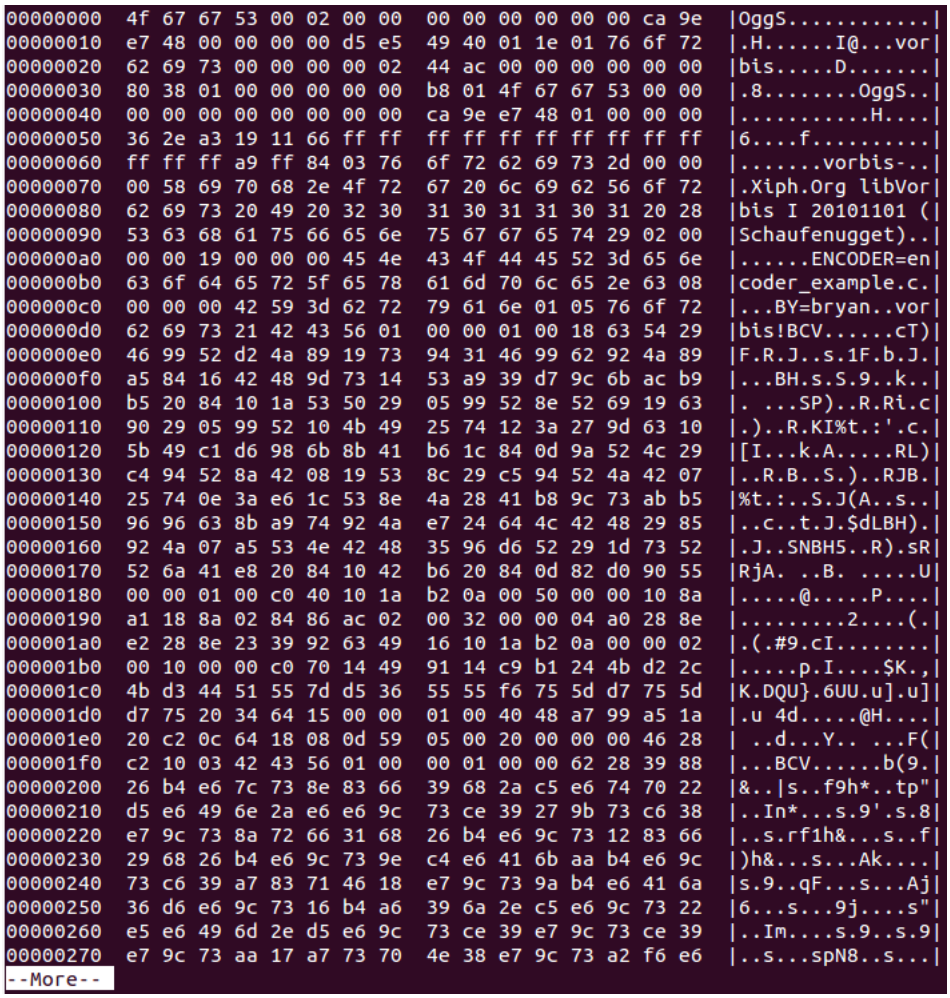

Figura 17: Execução da ferramenta hexdump.

3.5 Desenvolvimento do Editor

3.5.1 Construção do codificador

O estudo e o processo acima foi realizado para entender a estrutura do Ogg Vorbis e onde inserir os metadados e as marcações de conteúdo validando, assim, a possibilidade do uso do formato para a solução do problema. Para dar continuidade no estudo de viabilidade do formato Ogg Vorbis, foi desenvolvido um codificador em linguagem C. Como suporte, foram utilizadas as bibliotecas *libogg*, *libvorbis* e *libvorbisenc*. O *libvorbisenc* é responsável pela codificação. Para compilar o arquivo é necessário utilizar o seguinte comando:

$ gcc -o <nome_para_o_executável> <código_fonte> -logg -lvorbis -lvorbisenc

Para garantir que o processo de codificação realmente funcionasse, foi utilizado outro formato no processão de codificação de um formato Ogg Vorbis. O código desenvolvido pega o conteúdo sonoro de um formato WAVE e o codifica em Ogg Vorbis, com os seus dados comprimidos. Em outras palavras, o PCM do formato WAVE é decodificado e

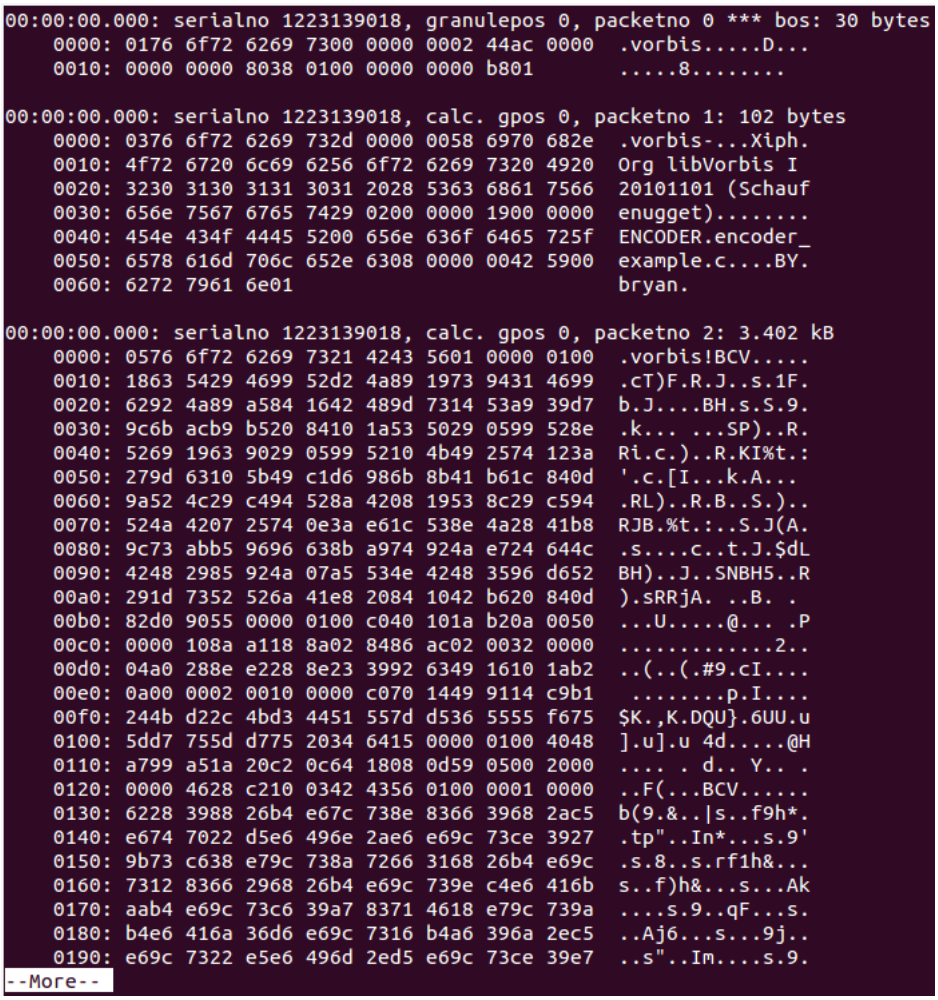

Figura 18: Execução da ferramenta oggz-dump.

posto em memória e, em seguida, os pacotes cabeçalhos do Ogg Vorbis são construídos. O PCM passa a ser inserido dentro do pacote de áudio finalizando o processo de decodificação. A ferramenta *oggz-dump* foi executada no arquivo gerado e após análise, os pacotes cabeçalhos, em tese, foram codificados corretamente. Para verificar se a integridade do arquivo não foi corrompida, utilizou-se o player Sound Exchange licenciado sob a GNU General Public License e distribuído por Chris Bagwell através (SOX, 2014). Este player possui uma interface de linha de comando e, ao utilizá-lo, era possível executar o som, este agora no formato Ogg, sem interrupção.

3.5.1.1 Inserção dos metadados

Como fundamentado teoricamente no item mais acima, o arquivo *.ogg possui um pacote onde é possível inserir comentários. O *comment packet* é o segundo pacote de cabeçalho da sequência de três que o Ogg Vorbis utiliza. Na Figura 18 ele aparece como o pacote número 1. O próximo passo então foi inserir, comentários referentes ao arquivo de áudio. Logo, o pacote de comentários do formato Ogg Vorbis foi utilizado para

inserção dos metadados e, comparando-o ao trabalho realizado (REIS, 2013), corresponde a estrutura META. Para este fim, o código desenvolvido em linguagem de programação C para a codificação foi modificado e este agora, além de pegar o conteúdo sonoro de um formato WAVE e o codificar em Ogg Vorbis com os dados comprimidos, ele também insere metadados no pacote. Para verificação da integridade do arquivo neste ponto do desenvolvimento, os mesmos passos utilizados no processo de codificação foram seguidos e as ferramentas *oggz-dump* e *sox* foram utilizadas.

3.5.1.2 Construção do pacote LGMK

Referente a estrutura LGMK (REIS, 2013), o formato Ogg Vorbis não possui suporte e se fez necessário a alteração de sua estrutura. Isso deveria ser feito, obviamente, sem que o arquivo fosse corrompido possibilitando sua execução em players comuns. Para tal finalidade, um novo pacote foi definido. A Figura 26 mostra quais são os campos que compõe o pacote.

Figura 19: Formato do pacote LGMK.

O campo referente ao **user_lgmks** é um array que armazena todas as marcações fornecidas pelo usuário e seu tamanho é ilimitado. O **lgmks_lengths** é um array responsável por armazenar o tamanho de cada marcação. A quantidade de marcações contidas no campo user_lgmks é armazenada no **lgmks**. Por último, o campo **user** contém informações sobre o usuário que fez as marcações. Uma nova estrutura foi definida e inserida dentro do formato sem, obviamente, corrompê-lo. Foi possível codificar e também decodificar as marcações de conteúdo com integridade.

Os três *header packets* definidos pelo Vorbis devem seguir a ordem disposta na Figura 20 ou o arquivo será corrompido. O pacote LGMK não é reconhecido pelo Vorbis pelos seguintes motivos:

1. O pacote possui um tipo diferente dos três tipos de cabeçalhos do Vorbis, desta forma ocorre erro no processo de decodificação.
2. A string de identificação do pacote é “lgmks” e o Vorbis não irá decodificar este pacote;

3. Quando o Vorbis for decodificar o pacote de áudio, o pacote LGMK SERÁ ignorado por ser um pacote não-áudio.

No entanto, a decodificação não é corrompida por conta da forma que foi estruturado o pacote e do local onde foi inserido. O pacote LGMK foi construído como sendo um pacote de cabeçalho semelhante aos *header packets* do *codec* Vorbis. Ou seja, ele possui um byte para a identificação do tipo de pacote de cabeçalho. A diferença está nos seis bytes subsequentes, onde a string que representa a identificação do pacote é "lgmks". O pacote foi inserido direto no formato recipiente Ogg logo após o terceiro pacote de cabeçalho do Voribs. A Figura 20 mostra como ficou organizado a estruturação do formato Ogg vorbis após inserção do pacote referente a marcação de conteúdo.

Como foi posto após o cabeçalho de configuração do Ogg Vorbis, no processo de decodificação padrão, o pacote LGMK tentará ser codificado como pacote de áudio e então será ignorado. Portanto, ao executar um player o arquivo Ogg Vorbis gerado é tocado normalmente.

3.5.2 Construção do decodificador

Para o desenvolvimento do código referente ao processo de decodificação do arquivo *.ogg foram utilizadas as bibliotecas *libvorbisfile*, *libvorbis* e *libogg*. A *libvorbisfile* oferece o suporte necessário voltado para a decodificação do arquivo tornando o processo mais simples. Para compilar o arquivo é necessário utilizar o seguinte comando:

$ gcc -o <nome_para_o_executável> <código_fonte> -logg -lvorbis -lvorbisfile

O código implementado decodifica os dados de cabeçalho e os imprime no terminal. Os dados PCM contidos do pacote de áudio são direcionados para um arquivo de saída.

3.5.2.1 Decodificação dos metadados

O metadados, uma vez codificados, precisavam ser decodificados e seu conteúdo recuperado em mémoria sem perda de dados. Para este fim, foi feito o uso da API *libvorbisfile* onde foi possível recuperar os metadados corretamente.

3.5.2.2 Decodificação do pacote LGMK

Como dito, o pacote é inserido diretamente no contêiner Ogg, ou seja, em nenhum momento é feito uso da estrutura vorbis para a inserção do pacote. Para o processo de decodificação do pacote LGMK, e por não fazer parte da estrutura Vorbis, a biblioteca *libvorbisfile* não oferece nenhum suporte e foi desconsiderada no processo de decodificação a partir deste ponto do projeto. Para alcançar o objetivo de decodificação, o uso das bibliotecas *libvorbis* e *logg* foram ainda mais efetivos. A *libvorbis* ainda foi utilizada

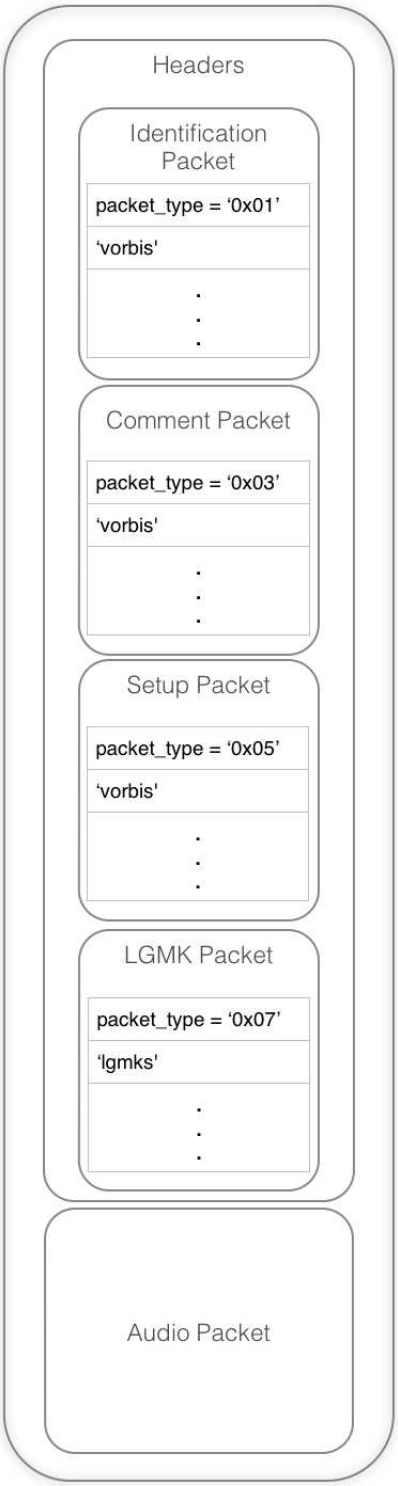

Figura 20: Estrutura Ogg Vorbis com o pacote LGMK inserido.

para decodificar a estrutura Vorbis, porém de uma forma mais "manual". Para o pacote LGMK foi utilizada apenas a biblioteca *logg*. Ao final, todo o arquivo foi decodificado corretamente.

3.5.3 Diagrama de Classe do Editor

Seguindo a arquitetura anterior, a Figura 21 apresenta o diagrama de classe para o Editor OGG desenvolvido conforme especificado nas sessões anteriores.

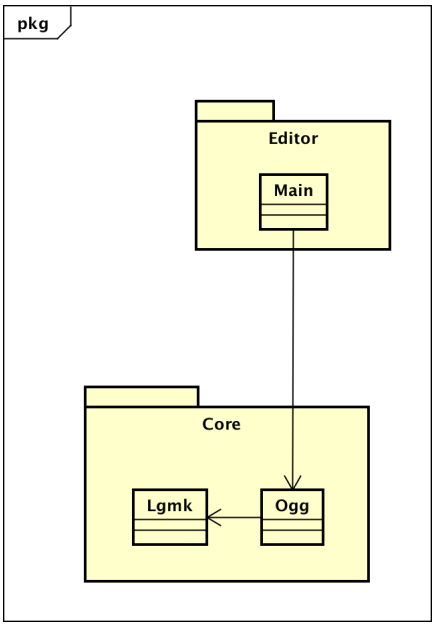

Figura 21: Diagrama de classe do Editor OGG.

Recorrendo aos serviços oferecidos pelo Core, a classe *Main* é responsável por abrir um arquivo OGG, construir o pacote de marcação e gerar, como saída, o formato com o pacote em sua estrutura. A classe *Ogg* é responsável por codificar e decodificar o formato. A construção do pacote de marcação está contido dentro da classe *Lgmk*. Tanto o pacote *Lgmk* quanto o pacote *Ogg* foram postos no Core do sistema pois estes serão necessários também para a execução do formato no Tocador.

3.6 Desenvolvimento do Tocador

O tocador foi inicialmente desenvolvido por (REIS, 2013) oferecendo suporte apenas aos formatos WAV e RAB. Desenvolvido em C++, o tocador faz uso do Qt para interfaceamento da apliacação e foi modificado originalmente para oferecer suporte ao novo formato comprimido especificado. A manutenção e evolução feita no Player está disseminada em boa parte do código visto que o Player passou a ser alimentado por este novo formato. O Tocador passou a fazer o uso da rotina libvorbis para descompressão dos dados e da API desenvolvida neste trabalho representada pelo Editor para a decodificação dos metadados e suas marcações. Referente a execução do áudio, para que fosse executado corretamente e os saltos fossem precisos, o cálculo para interpretação dos bytes do PCM do áudio decodificado foi reformulado. Na Figura 22 está representado a interface do Tocador proposta por (REIS, 2013):

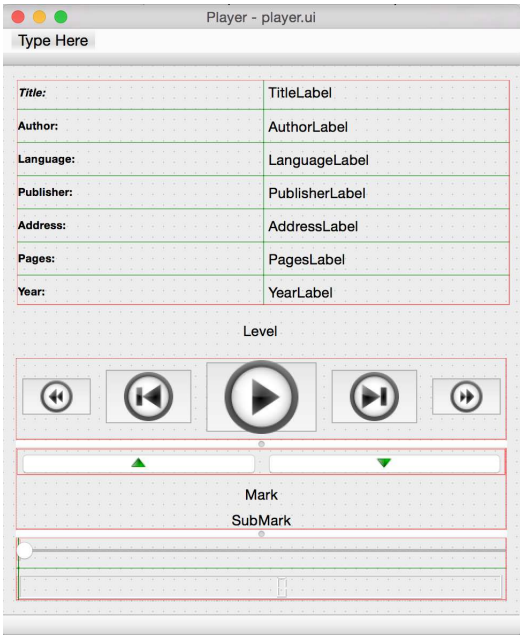

Figura 22: Layout do Tocador estruturado pela ferramenta Qt Desginer.

O Tocador é responsável por informar todos os dados contidos no formato. Os rótulos estáticos *Title, Author, Language, Publisher, Address, Pages e Year* indicam as informações referentes aos metadados do pacote *META* que, por sua vez, estão representados pelos *labels* ***TitleLabel, AuthorLabel, LanguageLabel, PublisherLabel, AddressLabel, PagesLabel e YearLabel***. O rótulo *Level* representa os dois possíveis níveis da marcação no que pode ser comparado aos capítulos e subcapítulos de um livro impresso tornando o acesso a informação mais preciso e coerente. *Mark* e *SubMark* são os rótulos responsáveis por informar a marcação e submarcação atual em que o áudio está contido. Ao final do layout tem-se uma barra para navegação manual do áudio e um visor digital que é atualizado constantemente mantendo a informação de tempo em segundos em que a execução do áudio se encontra.

Para fins de acessibilidade para pessoas que possuam limitação visual, o Tocador passou por melhorias de inteface e novas funcionalidades foram inseridas na aplicação. A elicitação dos novos requisitos, a análise destes requisitos e suas escolhas estão dispostos a seguir.

3.6.1 Elicitação de Requisitos de Acessibilidade

Para uma melhor precisão na escolha das funcionalidades que melhor contribuem para utilização de sistemas por pessoas com deficiência visual, 13 sistemas que contemplam este domínio foram analisados: *Blind's Music Player*, *Dolphin SuperNova Access Suite*, *JAWS*, *LianeTTS*, *MAGic*, *MECDAISY*, *NVDA Screen Reader*, *Projeto DOSVOX*, *Virtual Vision* e *Windows-Eyes*. Segundo o processo proposto por (sommerville), os requi-

sitos foram coletados, classificados e organizados em grupos coerentes. As funcionalidades comuns aos sistemas foram contados e priorizados considerando a funcionalidade de maior frequência como mais importantes:

Features	Softwares										Frequência
	Blind's Music Player	Dolphin SuperNova Access Suite	JAWS	LianeTTS	MAGic	MECDAISY	NVDA Screen Reader	Projeto DOSVOX	Virtual Vision	Windows-Eyes	
Sintetizador para leitura de tela		✓	✓	✓	✓		✓	✓	✓	✓	8
Contraste de Tela	✓	✓			✓	✓					4
Braille I/O		✓	✓			✓		✓			4
Sintetizador para informação de conteúdo	✓					✓	✓		✓		4
Sintetizador para ajuda	✓		✓			✓				✓	4
Destaque Visual	✓	✓			✓						3
Ampliador de Tela		✓			✓			✓			3
Treinamento			✓		✓					✓	3
OCR		✓	✓								2
Controlar Mouse Pelo Teclado		✓	✓								2
DAISY			✓			✓					2
Anúncio automático de texto sob o mouse							✓			✓	2
Fala de Caracteres ao pressionar tecla		✓									1
TOTAL	4/13	8/13	7/13	1/13	5/13	5/13	3/13	3/13	2/13	4/13	

Figura 23: Mapeamento das funcionalidades em comum.

A funcionalidade de suporte ao braille, apesar da frequência, não foi implementada por estar fora do escopo e propósito do trabalho aqui desenvolvido. As 7 funcionalidades de maior frequência foram implementadas, das quais são: sintetizador para leitura de tela, contraste de tela, sintetizador para informação de conteúdo, sintetizador para ajuda, destaque visual, ampliador de tela e anúncio automático de texto sob o mouse. Por intermédio destas *features* o cego ou a pessoa com baixa visão consegue acessar todas as informações referente ao *audiobook* que está sendo executado além de oferecer suporte de ajuda para utilização do Tocador.

3.6.2 User Stories

As *features* elicitadas estão representadas em *User Stories* e disponíveis no Apêndice A. Os Critérios de Aceitação para cada *User Story* estão disponíveis no Apêndice B.

3.6.3 Codificação

Após a elicitação, análise e definição dos requisitos o passo seguinte foi a implementação destes no sistema. Os três recursos básicos necessários para inserção das *features* de acessibilidade estão descritos nas sessões seguintes. A descrição de cada funcionalidade e o recurso e suporte que elas dispõem estão detalhados na seção Resultados Alcançados.

3.6.3.1 Sintetizador de Voz

Após a elicitação dos requisitos percebeu-se a necessidade do uso de recurso de voz para a aplicação. Para garantir que o recurso de voz funcionasse e estivesse bem integrado a aplicação foi utilizado a API QtSpeech, pois a ferramenta Qt já faz suporte ao desenvolvimento do Tocador. No entanto, foi utilizado uma versão modificada por Taodyne da biblioteca QtSpeech. Para uso do *tao-qt-speech* o código fonte acessível disponibilizado no github foi baixado e os comandos *make* e *make install* foram executados, nesta ordem, para instalação. Ela foi selecionada por fazer integração do suporte QtSpeech ao sintetizador de voz do próprio sistema em que a aplicação está sendo executada, sejam eles Mac OS X, Linux ou Windows.

3.6.3.2 Detectando Elementos de Interface

Para a correta implementação dos requisitos elicitados fez-se necessário a detectação e a diferenciação dos elementos dispostos na interface. Dependendo do elemento que estiver sob o mouse, o texto será ampliado ou os botões disponíveis serão falados e destacados visualmente. Para tal recurso o *EventFilter* é capaz de filtrar os objetos que estão sob o mouse fazendo diferenciação entre eles. O *EventFilter* deve ser instalado em cada objeto através do método *QObject::installEventFilter()*. Após isso, o método *eventFilter(QObject *object, QEvent *event)* fica responsável por escutar os objetos. Esta funcionalidade foi implementada na classe Ui_meta na camada *View* do Tocador como mostrado na Figura 24.

3.6.3.3 Recurso de Teclado

A definição das teclas funcionais para o Tocador também foram elicitadas a partir do reuso de requisitos na qual os requisitos de layout do teclado dos softwares citados foram analisados. Observou-se que não existe teclas específicas para cada funcionalidade de acessibilidade a pessoas com deficiência visual. No entanto, constatou-se que a tecla definida para dada funcionalidade é representada pela primeira letra do nome daquela funcionalidade, quando disponível. A fim de detectar quando uma tecla é pressionada e atribuir-lhe uma ação, os serviços das classes *QKeyEvent* e *QShortcut* do Qt foram utilizados por oferecerem o suporte necessário. A primeira coisa feita foi obter um referência de uma tecla por intermédio da classe *QKeySequence* que é responsável por encapsular uma tecla ou uma sequência delas para serem usadas com atalho no Tocador. Em seguida, a classe *QShortcut* cria um atalho que associa a tecla ao Tocador. Este atalho, por sua vez, foi conectado a um dado método que será executado no momento em que a tecla for pressionada. Estes passos foram implementados na controladora Player de acordo com a Figura 24.

O método *setControl(SoundCTRL *control)* habilita as teclas e as associa aos métodos da classe *SoundCTRL* da qual é responsável pelo controle do Tocador.

3.6.4 Diagrama de Classe do Tocador

Para a inserção dos requisitos de acessibilidade, dois métodos foram criados na controladora do Tocador. O primeiro deles é o *setKeyboardControl()*. Este método é responsável por criar os atalhos de teclado que executará os métodos de controle de execução do Tocador da classe SoundCTRL. O segundo método é o *setAccessibilitySupport()* cuja função é criar atalhos de teclado para os recursos visuais e do sintetizador de voz que estão definidos na classe Accessiblity. A diagrama de classe está representado na Figura 24.

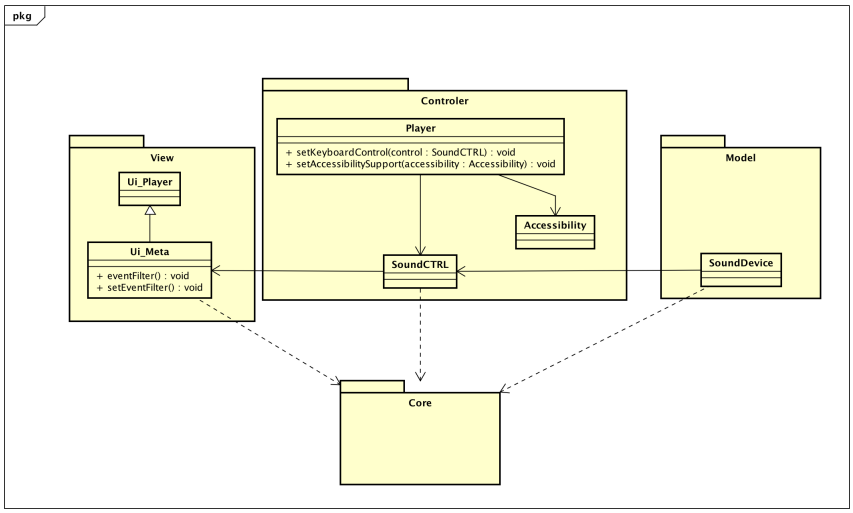

Figura 24: Diagrama de Classe do Tocador com os recursos de acessibilidade.

Para detecção do movimento do mouse e distinção entre os elementos sob ele os métodos *eventFilter()* e *setEventFilter()* foram implementados na classe Ui_Meta conforme dito na seção 3.6.3.2

3.6.5 Automação da Infraestrutura

Tradicionalmente, o desenvolvimento de software é marcado por uma série de processos manuais. Estes processos estão propensos a erros humanos cujo resultado ocasiona o aumento do custo de desenvolvimento e entrega da solução bem como a baixa qualidade do software entregue em decorrência destes erros. Ainda ocorre que o desenvolvimento e entrega de software flui em ambiente colaborativo entre desenvolvedores e também com o time operacional. Acontece que a comunicação e validação podem falhar em alguma parte do processo (BRAGA, 2015).

Diante do exposto e tendo em vista a automação do ambiente de desenvolvimento, teste e entrega, esta seção apresenta conceitos, ferramentas e técnicas que fornecem uma automatização que visa gerar confiabilidade, estabilidade e agilidade durante o processo, esta seção descreve o processo de Entrega Contínua assim como o detalhamento de seus subprocessos integração e implatanção contínua e pipeline de implantação.

Assim como o processo de validação e entrega, o ambiente de desenvolvimento foi automatizado. No decorrer do desenvolvimento do Tocador, o sistema operacional foi atualizado o que ocasionou na instabilidade do ambiente ao passo de não ser possível mais compilar o código. Por surgir a necessidade de se manter um ambiente estável, isolado e de fácil compartilhamento, iniciou-se então a configuração do Vagrant para o projeto. O processo de configuração bem como os problemas encontrados após atualização do sistema operacional e suas respectivas soluções estão descritos abaixo:

Vagrant Box: instalação da box *chad-thompson/ubuntu-trusty64-gui* que contém o ubuntu 14.04 com interface gráfica;

Instalação do Qt: instalação do Qt através dos seguintes comandos:

$ wget http://download.qt.io/official_releases/qt/5.7/5.7.0/qt-opensource-linux-x64-5.7.0.run

$ chmod +x qt-opensource-linux-x64-5.7.0.run

$ ./qt-opensource-linux-x64-5.7.0.run

Instalação do qmake: executar o comando *$ sudo apt-get install qt4-qmake libqt4-dev*;

Instalação do SDL 1.2: instalação do SDL a partir do comando *$ sudo apt-get install libsdl1.2-dev libsdl-image1.2-dev libsdl-mixer1.2-dev libsdl-ttf2.0-dev*

Uso de bibliotecas do c++11: adicionou-se as seguintes linhas ao arquivo Player.pro para uso da biblioteca thread:

QMAKE_CXXFLAGS += -std=gnu++0x -pthread

QMAKE_CFLAGS += -std=gnu++0x -pthread

Problema com a main(): isso ocorria porque não era encontrada a main() para o ponto de entrada da aplicação. A função utilizada era SDL_main() no OS X, para o Ubuntu, portanto, a função foi renomeada para main().

Sintetizador de voz QtSpeech: execução do comando para instalação *$ sudo apt-get install festival-dev speech-tools*

Instalação da ferramenta Sox: instalando a partir do comando *$sudo apt-get install sox*

Problema com data de criação de arquivos gerados: como os arquivos são compartilhados entre dois sistemas, havia problema de sincronismo de relógio. Para suporte a solução do problema, foi instalada a ferramenta ntp. Após, o comando $ touch * foi executado para correção dos arquivos e os relógios foram sincronizados ao parar e inicializar o serviço ntp.

Para inicializar a máquina virtual a partir do Vagrant, basta executar o comando *vagrant up –provision*, onde o parâmetro *up* irá subir a máquina virtual e a flag *–provision* executará o script descrito no arquivo Vagrantfile. O diretório atual é compartilhado com a máquina virtual em tempo real onde as alterações são sensibilizadas em ambos os sistemas. Para isso, o Vagrant foi configurado com compartilhamento de pastas usando NFS possibilitando a edição dos arquivos de implementanção na máquina real e compilacão do projeto na máquina virtual.

De forma semelhante ao Vagrant, o Travis sobe uma máquina virtual para então realizar as tarefas que lhe foram propostas. Assim, um script foi definido para que ele fosse capaz de montar o ambiente virtual com as ferramentas necessárias para compilação, teste e implantação do Tocador. Este mesmo script contém os passos definidos para o pipeline de implantação.

O Codeclimate por si não oferece suporte para c/c++ e, com isso, fez-se necessária a instalação do plugin Cppcheck. O Cppcheck tem um forte foco na detecção de comportamento indefinido, dos quais são: ponteiros mortos, divisão por zero, estouros de inteiro, operandos de deslocamento de bit inválidos, conversões inválidas, uso inválido do STL, gerenciamento de memória, dereferências de ponteiro nulo, verificação fora dos limites, variáveis não inicializadas e escrevendo dados constantes.

3.6.5.1 Entrega Contínua

Entrega contínua é uma prática que garante a entrega de software da equipe de desenvolvedores para o ambiente de produção em um processo confiável, previsível, visível e o mais automatizado possível, com riscos quantificáveis e bem entendidos (HUMBLE; FARLEY, 2010). A prática automatiza todo o sistema de entrega (build, implantação, teste e liberação) reduzindo o tempo e os riscos associados à entrega de novas versões de softwares. Assim, a Entrega Contínua é a junção da Integração e Implantação Contínua.

3.6.5.2 Integração Contínua

Com surgimento na metodologia ágil XP, o termo está difundido em diversas metodologias e se define na prática na entrega do código alterado e/ou desenvolvido na mesma frequência que as funcionalidades são desenvolvidas. Para tal, a integração contínua ocorre

diversas vezes ao dia e seu objetivo principal consiste em verificar se as alterações realizadas no projeto não produziram erros no projeto já existente (BECK; FOWLER, 2001). Sendo assim, a Integração Contínua permite a integração das frequentes alterações parciais ao código principal por um processo automatizado de build e testes com garantias e, por fim, notificações para conhecimento do time de desenvolvimento sobre a operação.

3.6.5.3 Implantação Contínua

Para fazer a Implantação Contínua, deve-se estar fazendo a Entrega Contínua. A Entrega Contínua significa que você é capaz de realizar implantações frequentes, mas pode optar por não fazê-lo. Portanto, a Implantação Contínua é o estágio final da entrega contínua por onde todas as alterações que passam pelo pipeline são automaticamente colocadas em produção (HUMBLE; FARLEY, 2010).

3.6.5.4 Pipeline de Implantação

De modo geral, a implementação do pipeline de implatancão tem por finalidade detectar quaisquer alterações que levem a problemas na produção, sendo uma manifestação automatizada do processo de levar o software de controle de versão até os usuários. Em outras palavras, o pipeline é contruído em estágios que evidenciam as mudanças de construção do software que passam por um processo de compilação, por estágios de testes e implantação. O pipeline de implantação deve permitir a colaboração entre os vários grupos envolvidos no fornecimento de software e fornecer visibilidade a todos sobre o fluxo de alterações no sistema fornecendo assim *feedback* o mais rápido possível (HUMBLE; FARLEY, 2010).

Normalmente, o primeiro estágio de um pipeline realiza uma compilação que fornecerá um binário para os estágios seguintes. Os estágios posteriores podem incluir verificações manuais ou automáticas, como quaisquer testes que podem ser ou não ser automtatizados. A implantação na produção é geralmente o estágio final de um pipeline (HUMBLE; FARLEY, 2010). Partindo deste pressuposto, à medida que o software passa em cada estágio ele vai se tornando mais confiável e mais próximo de ir à produção. Os testes que falham em algum estágio descarta a versão candidata e o processo de *feedback* a respeito da falha é obtido o mais rápido possível.

Assumindo o processo básico de um pipeline de implantação proposto por Humble Jez, Farley David (HUMBLE; FARLEY, 2010) com modificações para adequação ao escopo deste trabalho, abaixo está descrito os estágios que compõe o pipeline de implantação a ser utilizado no Tocador. O processo inicia quando:

1. Controle de versão: o código é submetido ao GitHub pela equipe de desenvolvimento e dispara o processo de pipeline;

2. Análise estática de código: o código submetido passa por uma análise e as métricas de qualidade são colhidas;

3. Provisionamento automático: ocorre a instalação dos pacotes necessários a execução da aplicação;

4. Automatização de builds de teste: gerar executável em modo teste para que os testes unitários sejam executados;

5. Automatização de testes: executação dos testes unitários;

6. Automatização de builds executáveis: gerar executável em modo debug para criação do binário a ser implantado;

7. Automatização de deploy: implantação do binário em um ambiente simulado de produção. Disponibilização do binário para download.

Em caso de ocorrência de falha, o histórico do processo do pipeline executado é registrado e um e-mail de notificação é enviado para a equipe de desenvolvimento. Durante o pipeline para implantação o passo seguinte só iniciará se o anterior for bem sucedido.

4 Resultados Alcançados

4.1 O Editor OGG

O Editor de *audiobooks* gera com sucesso um formato Ogg Vorbis com todos os metadados inseridos no pacote Vorbis de comentários e as marcações de conteúdo contidas no pacote LGMK construído. Mesmo com sua estrutura alterada, o formato *.ogg pode ser executado em players com suporte a estes arquivos como, a exemplo, o *sox* ou Rhythmbox.

A efeito de comparação ao (REIS, 2013), cujo trabalho desenvolveu um formato parecido que não faz uso da compressão de dados, também foi utilizado o Hino Nacional Brasileiro para simular um *audiobook*. O arquivo original no formato WAV possui 3 minutos e 41 segundos de duração e ocupa 42,5 MB de espaço em memória. Este arquivo foi carregado pelo Editor e em seguida foram inseridos metadados conforme mostra Tabela 1.

Tabela 1: Metadados inseridos no *comment header*.

Nome	**Valor**
Título	Hino Nacional Brasileiro
Autor	Joaquim Osório & Francisco Manuel
Idioma	Português
Editora	
Local de Criação	Brasil
Número de Páginas	0
Ano de Criação	1822

Para verificação dos metadados inseridos no pacote de comentários do formato Ogg, a Figura 25 apresenta a decodificação do arquivo utilizando a ferramenta ogg-dump. O pacote de comentários está identificado como pacote de número 1 e nele é possível notar que os metadados foram inseridos corretamente.

Sobre as marcações de conteúdo, a Tabela 2 lista as informações inseridas no pacote LGMK. Cada item da tabela representa uma marcação composta por uma legenda e o tempo em que se deve estar posicionada.

A verificação da inserção dos marcadores de conteúdo no pacote LGMK pode ser visualizado usando a ferramenta oggz-dump. A Figura 26 mostra o pacote LGMK que pode ser identificado pelo número de pacote 3 e sua descrição evidencia os marcadores de conteúdo que o compõe.

O pacote Ogg Vorbis gerado possui apenas 1,84MB e não mais 42,5MB como o

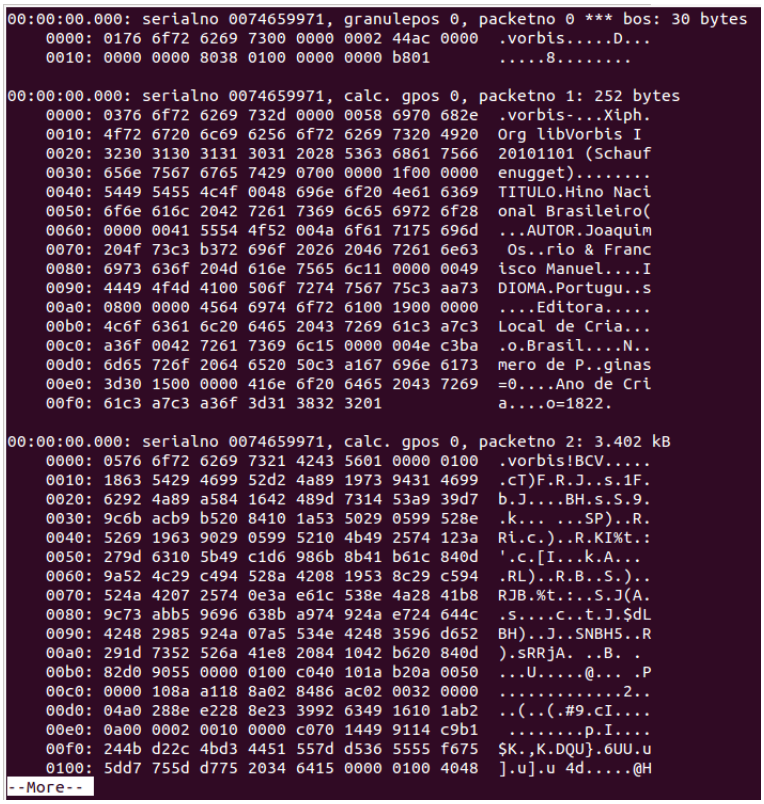

Figura 25: Utilização do oggz-dump - pacote de comentários.

arquivo original. Houve uma redução considerável devido ao algoritmo de compressão do *codec* Vorbis. O arquivo gerado é executado corretamente do início ao fim sem interrupções. Também foi feito um teste para verificar se o pacote LGMK não iria corromper o arquivo Ogg Voribs ao inserir um número considerável de marcacões, um total de 22 milhões de marcações. O arquivo final ficou com 340MB de tamanho, no entanto, o player conseguiu executá-lo normalmente.

4.2 O Tocador

O Tocador desenvolvido é capaz de executar o novo formato gerado pelo Editor OGG e, através das marcações contidas no formato, navegar pelo conteúdo do áudio facilmente. O Tocador está representado na Figura 27.

O Tocador carrega os metadados referente ao arquivo informando ao usuário sobre qual *audiobook* está sendo executado assim como mantém sempre atualizado as informações das marcações e o tempo da execução do *audiobook*. A execução do áudio também pode ser controlada pelo *slider*. O *audiobook* pode ser pausado ou executado a qualquer momento que o usuário desejar. Os botões alinhados a esquerda e direita do botão de *play/pause* são responsáveis por realizar os saltos para as marcações da seguinte forma: os mais próximos saltam para a marcação mais próxima de mesmo nível enquanto os

Tabela 2: Marcações inseridas no *LGMK header*.

Nome da Marcação	**Tempo(s)**
Parte 1	0
Parte 2	103
Introdução 1	0
Primeira Estrofe	28
Segunda Estrofe	44
Terceira Estrofe	60
Quarta Estrofe	64
Quinta Estrofe	79
Sexta Estrofe	91
Introdução 2	103
Sétima Estrofe	132
Oitava Estrofe	148
Nona Estrofe	164
Décima Estrofe	168
Décima Primeira Estrofe	183
Décima Segunda Estrofe	195
Décima Terceira Estrofe	201
Final	206

laterais saltam para a marcação mais próximo seguindo a linha do tempo. Ou seja, as duas formas de navegação segue o tempo de forma sequencial (dependendo da direção: crescente ou decrescente) mas um navega considerando apenas as marcações de mesmo nível e o outro todas as marcações. As funcionalidades que tornam o Tocador acessível também as pessoas com deficiência visual estão expostas nas seções a seguir.

4.2.1 Contraste de tela

Conforme fundamentado na seção 2.4.1, as cores foram definidas a partir do modelo que alcança diversos tipos de diagnósticos de baixa visão. Assim sendo, o fundo preto e as letras amareladas foram aplicadas por minimizar o problema da maioria dos casos atingindo resultados satisfatórios. Este requisito proporciona ao usuário um contraste alternativo de acordo com a pesquisa elicitada. A Figura 28 mostra o Tocador com a aplicação do modelo de cores.

4.2.2 Sintetizador para informação de conteúdo

O recurso para leitura da informação do conteúdo é iniciado quando a tecla *i* é pressionada. A informação do conteúdo são os metadados do *audiobook* e, quando solicitado pelo usuário, o Tocador irá ler as seguintes informações por meio do sintetizador de voz: título, autor, idioma, editora, endereço, páginas e ano. Mais informações podem ser

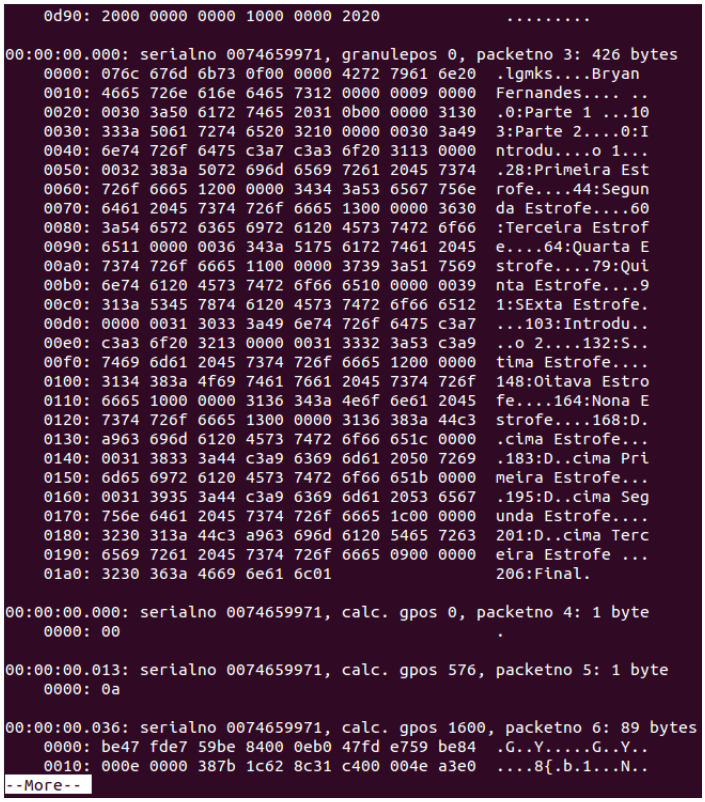

Figura 26: Utilização do oggz-dump - pacote LGMK.

lidas pelo sintetizador de voz através das teclas m e n que informam, respectivamente, as marcação atual e o nível da navegação das marcações.

4.2.3 Sintetizador para ajuda

Esta funcionalidade é iniciada quando o usuário pressiona a tecla h. Por conseguinte, o sintetizador de voz procederá lendo para o usuário quais são os recursos de atalho do teclado disponíves na aplicação.

4.2.4 Destaque visual

Os botões da aplicação não são destacados visualmente e, para tal, esta funcionalidade foi implementada. Para as pessoas com baixa visão, um botão será destacado com uma borda colorida enquanto este estiver sob o ponteiro do mouse favorecendo a identificação do botão que poderá ser pressionado, como exemplificado na Figura 29.

Para cada funcionalidade um cor para destaque foi definida como listado abaixo:

- **Verde**: botão de play/pause;
- **Azul**: botões que saltam para marcações próximas de mesmo nível;
- **Vermelho**: botões que saltam para marcações próximas independente do nível;

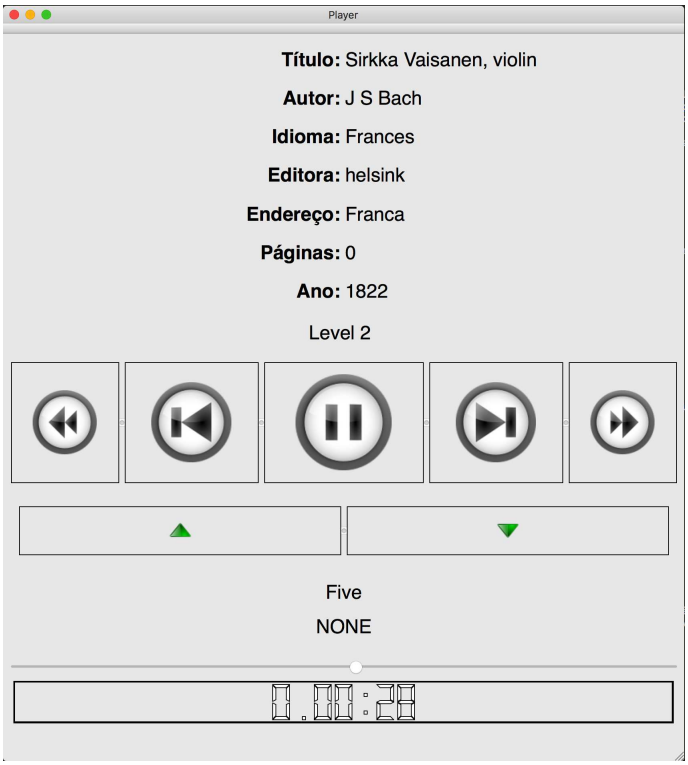

Figura 27: Execução de um arquivo de teste gerado pelo Editor.

- **Laranja**: botões que alteram nível de navegação das marcações.

4.2.5 Ampliador de tela

Todo o conteúdo de texto presente na tela do Tocador será ampliado consideravelmente para que o texto ressalte e a leitura seja facilitada aos usuários que possuem baixa visão. O texto permanecerá ampliado enquanto o mouse estiver sobre o texto conforme podemos ver na Figura 30.

4.2.6 Anúncio automático de texto sob o mouse

Este requisito reutilizado foi implementado nos softwares analisados para ler interfaces gráficas desconhecidas como, por exemplo, *websites*. Como a interface do Tocador é conhecida e já temos suporte de retorno de voz para as informações contidas nela por meio do atalho do teclado, esta *feature* foi adaptada para ter um melhor suporte a usabilidade. Para este fim, o anúncio automático sucederá quando o mouse estiver sobre um botão. Assim sendo, o Tocador oferece suporte, por intermédio desta fucionalidade, de anúncio automático de botão sob o mouse onde este será pronunciado pelo sintetizador de voz. À vista disso, o usuário saberá o botão que estará acionando antes mesmo do clique do mouse.

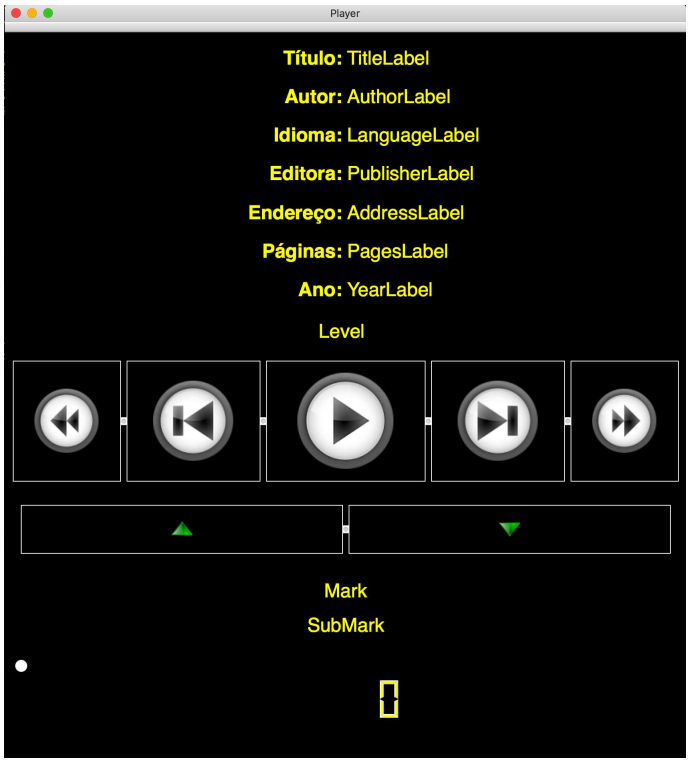

Figura 28: Contraste de tela alternativo.

4.2.7 Sintetizador para leitura de tela

Diferente dos demais requisitos de acessibilidade apresentandos neste trabalho, este requisito não será iniciado pelo usuário mas sim pelo próprio Tocador. Esta *feature* é responsável por informar o usuário sobre as mudanças de estado da interface gráfica como, por exemplo, o *audiobook* passar de uma marcação a outra durante a execução ou ainda quando houver mudança de nível da marcação.

4.2.8 Melhorias de Interface

Para as especificações do sistema proposto por (REIS, 2013), a interface atende aos requisitos. Contudo, não são suficientes para que pessoas com baixa visão façam distinção e tenham percepção do que está sendo mostrado em tela. Atendendo aos requisitos, o Tocador foi redimensionado por completo, os *labels* passaram a estar centralizados e os botões agora possuem mesma dimensão. Em tempo de execução, os botões são destacados, as informações textuais são ampliadas e a contraste de tela pode ser alterado.

4.2.9 Uso do Teclado

O uso do Tocador não se dá apenas pela interface gráfica através do uso do mouse, pelo teclado também somos capazes de utilizar todos os comandos disponíveis acessíveis para navegação e controle de execução do conteúdo de áudio gravado. O recurso de ajuda

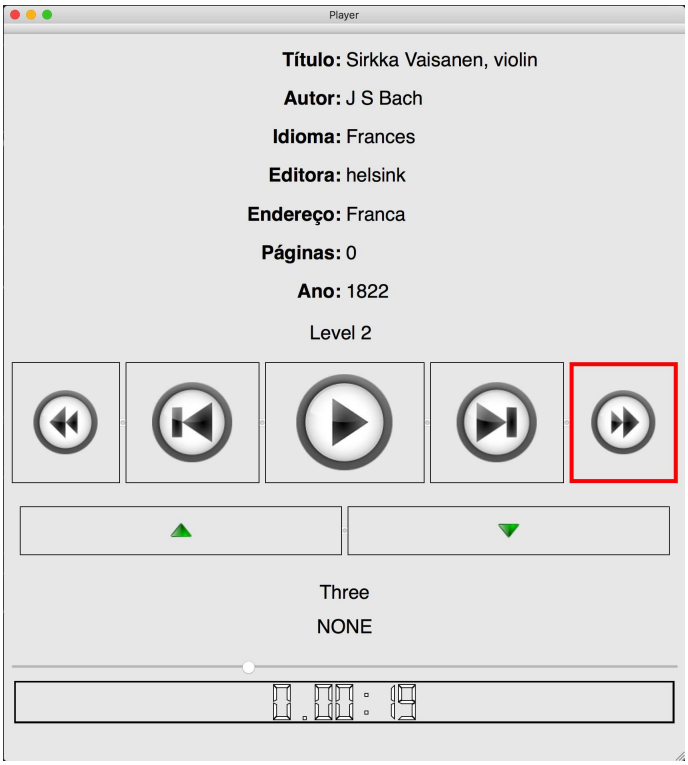

Figura 29: Destacando botão sob o mouse.

é comumente representada pela tecla *h* fazendo referência ao nome inglês *help*. Para navegação no conteúdo do *audiobook*, as teclas direcionais são utilizadas. Os comandos pelo teclado responsáveis pelo controle da execução do *audiobook* são:

- **Play/Pause**: o comando pode ser executado alternativamente pela tecla de espaço;
- **Pular uma marcação a frente**: esta funcionalidade do player pode ser acionada através da tecla de seta para a direita;
- **Voltar uma marcação**: esta funcionalidade do player pode ser acionada através da tecla de seta para a esquerda;
- **Subir a marcação para o nível 1**: o comando pode ser acionado quando pressionada a tecla de seta para cima. Esta funcionalidade foi pensada com o intuito de contemplar a referência de capítulos contidas nos livros impressos;
- **Descer a marcação para o nível 2**: esta funcionalidade do player pode ser acionada através da tecla de seta para a baixo. Esta funcionalidade do player visa contemplar a referência para os subcapítulos como encontrado nos livros impressos, tornando a busca de informação ainda mais otimizada.

O Tocador possue outras teclas de atalhos que inicializam os recursos de acessibilidade. As respectivas teclas e seus atalhos funcionais estão detalhados abaixo:

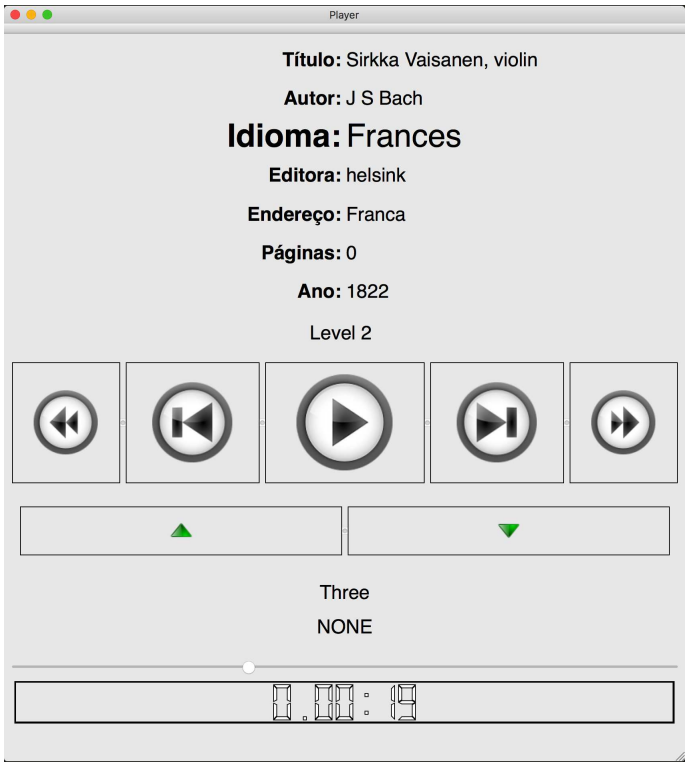

Figura 30: Funcionalidade ampliador de tela do Tocador.

- **tecla *i***: quando pressionada, o sintetizador de voz irá ler os metadados do *audiobook* de acordo com a seção Sintetizador para informação de conteúdo;

- **tecla *h***: ao pressionar esta tecla, o sintetizador de voz irá pronunciar ao usuário os atalhos disponíveis como mencionado na seção Sintetizador para ajuda;

- **tecla *c***: o usuário poderá alternar entre dois contrastes de telas diferentes a cada vez que esta tecla for pressionada em acordo com a seção Contraste de tela;

- **tecla *m***: presionando esta tecla o sintetizador de voz irá ler a marcação atual em que a execução do *audiobook* se encontra como dito na seção Sintetizador para informação de conteúdo;

- **tecla *n***: e, por último, o usuário terá um retorno audível sobre qual o nível atual em que está o salto das marcações pressionando esta tecla conforme falado na seção Sintetizador para informação de conteúdo.

O Tocador foi testado nos sistemas operacionais Ubuntu e Mac OS X. Nos dois ambientes o Player não apresentou erro durante a sua execução. Importante ressaltar que o Editor também é funcional nestes dois sistemas operacionais.

4.3 A Engenharia de Produto

O processo de desenvolvimento do Tocador foi automatizado. Na parte da implementação do software, o script elaborado e a box selecionada são capazes de montar um ambiente estável para colaboração com o desenvolvimento do software utilizando-se do VirturalBox através da ferramenta Vagrant. O script para montagem do ambiente é necessário rodar somente a primeira vez. Após, a máquina precisou apenas ser iniciada. Do momento inicial ao uso subsequente da máquina virtual gerenciada pelo Vagrant o mesmo não apresentou falhas durante o uso ou instabilidade que poderia tornar o ambiente instável.

O processo de Entrega Contínua inicia-se quando um commit é realizado no repositório GitHub que atua no controle de versão do projeto. Com isso, inicia-se também o pipeline de implantação. No repositório é possível visualizar o histórico de commits que ocasionaram falha em algum ponto do pipeline ou sucesso em todo o processo. Se não ocorrer falha durante os testes, o Pipeline de Implantação especificado é capaz de entregar o executável ao final do processo em ambiente de produção simulado. Os demais estágios do pipeline que compõe o processo de integração e implantação contínua estão descritas a seguir.

A cada novo commit efetuado no repositório GitHub, o Codeclimate, por meio do plugin Cppcheck, analisou estaticamente todo o código fonte e registrou, quando ocorrido, as *issues* conforme mencionado em Automação da Infraestrutura. Para potenciais erros do projeto, o Codeclimate notifica a equipe de desenvolvimento sobre a *issue* de maior risco. Para visualização dos dados coletados, acesse CodeClimate.

Paralelo a análise estática, o Travis passa a cuidar do restante do processo do pipeline de implantação. Como um próximo passo, o Travis foi capaz de configurar o ambiente corretamente de maneira a preparar o ambiente virtual para compilação, teste e implantação do Tocador. Uma vez que o provisionamento do ambiente é realizado, o projeto é compilado em modo teste com suporte do QtTest. Os testes automatizados são executados e em caso de sucesso, o Travis segue para a construção do binário a ser implantando. Por fim, o binário é disponibilizado para download e assim o pipeline de implantação é concluído. Vale mencionar que ao término de cada execução, o Travis cuida em notificar a equipe de desenvolvimento sobre o resultado da execução. Durante a elaboração do trabalho, o script que direciona o Travis para os passos do pipeline a serem executados não apresentou falhas ou inconsistência durante o processo. Para visualização dos *logs* acesse Travis CI.

5 Conclusão

Este trabalho demonstrou que a acessibilidade dificilmente é lembrada e faz parte do processo do desenvolvimento de uma solução tecnológica. Este fato é ainda mais evidente quando a solução não é estritamente focada para este público alvo. Tornar um sistema acessível para pessoas que possuam deficiência visual não é criar algo completamente novo mas é sabermos usar o que já existe e manejarmos de forma combinativa e integrada todo o aparato tecnológico disponível como suporte ao desenvolvimento de sistemas acessíveis.

No início de um processo de desenvolvimento de software, requisitos de acessiblidade devem ser pensados e postos em prática. Por conta das restrições de acesso e uso de softwares existentes, muitas pessoas não conseguem fazer uso da solução tecnológica. O formato OGG, do qual foi derivado o formato especificado neste trabalho, é de código aberto e seus desenvolvedores não só permitem como incentivam a comunidade para contribuir com o formato. A partir desta plataforma, uma novo direcionamento é dado ao formato OGG. Como conclusão deste trabalho, novos pacotes foram criados e inseridos na estrutura do formato de áudio de forma a proporcionar a inserção de metadados e marcações do conteúdo do pacote de áudio. Com o uso deste novo modelo, além do áudio é possível adicionar informações referentes ao *audiobook* (tais como: título, ano, editora, autor, entre outros) e navegar por capítulos e marcações. Deste modo, ao compartilhar o arquivo de *audiobook*, o usuário não só terá o áudio mas a informação decorrente dele.

Com os requisitos necessários para tornar um Player acessível aos deficientes visuais, observou-se que o interfaceamento da aplicação foi completamente replanejado. Não somente a tela, mas o teclado recebeu novas funcionalidades também. Devido a tais modificações, constatou-se que os recursos de acessibilidade definem a interface e interação do Player com o usuário, de modo a garantir que o deficiente visual possa ter o conhecimento de todas as informações disponíveis.

Espera-se que este trabalho contribua para uma nova forma de leitura de livros didáticos de maneira agradável e acessível para um número maior de pessoas dos que os livros impressos proporiconam. O arquivo gerado pelo Editor, faz uso da compressão de dados tornando seu tamanho reduzido o que facilita a disseminação dos *audiobooks* pela Internet. O formato é *open source* o que não trás um custo adicional para as editoras, diminuindo, consequentemente, o custo dos *audiobooks*. Em um ambiente escolar, por exemplo, o ensino pode ser maximizado e pessoas com deficiência visual podem atingir seu potencial e fazerem sua plena contribuição a sociedade.

5.1 Trabalhos Futuros

Como continuação deste trabalho, o próximo passo seria a avaliação da solução tecnológica em um ambiente real. Uma metodologia de pesquisa de campo definida e utilizada como suporte a avaliação do uso da ferramenta por pessoas com deficiência visual. Os dados coletados e analisados podem gerar um relatório conclusivo sobre a proposta apontando seu desempenho como material didático de apoio aos deficientes visuais.

Referências

ACM. *ACM Digital Library*: Site. 2014. Disponível em: <http://dl.acm.org>. Acesso em: 12 nov. 2014. Citado na página 47.

ALECRIM, E. Mp3. Info Wester, 2006. Disponível em: <http://www.infowester.com/histomptres.php>. Acesso em: 05 nov. 2014. Citado na página 36.

ALUNOS cegos enfrentam dificuldades em escolas brasileiras: Site. 2009. Disponível em: <http://noticias.terra.com.br/educacao/interna/0,,OI3593724-EI8266,00-Alunos+cegos+enfrentam+dificuldades+em+escolas+brasileiras.html>. Acesso em: 23 maio. 2015. Citado na página 30.

AUDIBLE. *Home page*: Site. 2014. Disponível em: <http://www.audible.com>. Acesso em: 27 out. 2014. Citado na página 26.

BECK, K.; FOWLER, M. *Planning Extreme Programming*. Addison-Wesley, 2001. (The XP series). ISBN 9780201710915. Disponível em: <https://books.google.com.br/books?id=u13hVoYVZa8C>. Citado 2 vezes nas páginas 44 e 64.

BECKER, F.: Entendendo o mp3 (cbr, abr, vbr). 2014. Disponível em: <http://www.midideejay.com/2009/03/tutorial-mp3-cbr-abr-vbr.html>. Acesso em: 15 nov. 2014. Citado na página 41.

BLANCK, P. E-books must be accessible, and that means audio. 2010. Disponível em: <http://chronicle.com/article/E-Books-Must-Be-Accessible/64518/>. Acesso em: 14 mar. 2015. Citado na página 21.

BRAGA, F. A. M. Um panorama sobre o uso de práticas devops nas indústrias de software. 2015. Disponível em: <https://repositorio.ufpe.br/handle/123456789/15989>. Acesso em: 15 nov. 2018. Citado na página 61.

BURKEY, M. Accessible audiobooks for the print disabled. 2013. Disponível em: <http://www.booklistonline.com>. Acesso em: 14 mar. 2015. Citado 2 vezes nas páginas 21 e 27.

CARTILHA do Censo 2010: Site. 2015. Disponível em: <http://www.pessoacomdeficiencia.gov.br/app/sites/default/files/publicacoes/cartilha-censo-2010-pessoas-com-deficienciareduzido.pdf>. Acesso em: 23 maio. 2015. Citado na página 29.

COHN, M. *User Stories Applied: For Agile Software Development (Adobe Reader)*. Pearson Education, 2004. (Addison-Wesley Signature Series (Beck)). ISBN 9780132702645. Disponível em: <https://books.google.com.br/books?id=DHZP_YL3FxYC>. Citado na página 44.

DEPUTADOS, C. dos. *Como lidar com as pessoas com deficiência*: Site. Disponível em: <http://www2.camara.leg.br/responsabilidade-social/acessibilidade/Como-lidar.html>. Acesso em: 28 out. 2015. Citado na página 25.

DW. Audiolivro vira moda na alemanha. DW, Deutsche Welle, 2004. Disponível em: <http://dw.de/p/4wvJ>. Acesso em: 27 out. 2014. Citado na página 26.

ERROR. *MPEG Audio Compression Basics*: Site. 1999. Disponível em: <http://www.mpgedit.org/mpgedit/mpeg_format/mpeghdr.htm>. Acesso em: 03 nov. 2014. Citado na página 35.

FALADA, U. *Home page*: Site. 2004. Disponível em: <http://www.universidadefalada.com.br>. Acesso em: 02 nov. 2014. Citado na página 27.

FARIAS, S. C. O audiolivro e sua contribuição no processo de disseminação de informações e na inclusão social. Revista Digital de Biblioteconomia e Ciência da Informação, 2012. Disponível em: <http://www.sbu.unicamp.br/seer/ojs/index.php/rbci/article/view/529>. Citado na página 26.

GOOGLE. *Google Acadêmico*: Site. 2014. Disponível em: <http://scholar.google.com.br>. Acesso em: 15 nov. 2014. Citado na página 47.

GRENNING, J. W. *Test Driven Development for Embedded C*. Pragmatic Bookshelf, 2011. (Pragmatic Bookshelf Series). ISBN 9781934356623. Disponível em: <https://books.google.com.br/books?id=QuUBRQAACAAJ>. Citado na página 44.

GRIFFITH, A. Ensinando cores contrastantes na arte. Disponível em: <http://www.ehow.com.br/ensinando-cores-contrastantes-info_388669/>. Acesso em: 21 nov. 2015. Citado na página 31.

HUMBLE, J.; FARLEY, D. *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation (Adobe Reader)*. Pearson Education, 2010. (Addison-Wesley Signature Series (Fowler)). ISBN 9780321670229. Disponível em: <https://books.google.com.br/books?id=6ADDuzere-YC>. Citado 2 vezes nas páginas 63 e 64.

JESUS, P. S. *Inclusão Ponto a Ponto: a inclusão social da pessoa com deficiência visual através da produção escrita*. Monografia (Anais), 2007. Citado na página 29.

KULPA, C. C.; TEIXEIRA, F. G.; SILVA, R. P. Um modelo de cores na usabilidade das interfaces computacionais para os deficientes de baixa visão. 2010. Disponível em: <http://www.pgdesign.ufrgs.br/designetecnologia/index.php/det/article/viewFile/8/7>. Acesso em: 21 nov. 2015. Citado na página 31.

LEGAL, B. *Convenção sobre os Direitos das Pessoas com Deficiência*: Site. 2006. Disponível em: <http://www.bengalalegal.com/convencao>. Acesso em: 28 out. 2015. Citado na página 25.

MALCZEWSKI, B. Multi-tasker's dream. 2012. Disponível em: <http://reviews.libraryjournal.com/2012/04/media/audio/multi-taskers-dream-as-new-formats-emerge-audiobooks-seem-poised-to-expand-their-reach-but-challenge
Acesso em: 18 mar. 2015. Citado na página 28.

MATOS, S. R. Educação, cidadania e esclusão à luz da educação especial: retrato da teoria e da vivência. 2003. Disponível em: <http://www.ibc.gov.br/Nucleus/index.php?catid=4&itemid=46>. Acesso em: 21 mar. 2015. Citado na página 29.

MURATNKONAR. *Audio Interchange File Format*: Site. 2014. Disponível em: <http://www.muratnkonar.com/aiff/index.html>. Acesso em: 03 nov. 2014. Citado na página 34.

NETO, L. Livros para ouvir. PublishNews, 2014. Disponível em: <http://www.publishnews.com.br/telas/noticias/detalhes.aspx?id=79097>. Acesso em: 11 nov. 2014. Citado na página 27.

PALLETA, F. A. C.; WATANABE, E. T. Y.; PENILHA, D. F. Audiolivro: inovações tecnológicas, tendências e divulgação. 2008. Citado na página 26.

PALMER, A. A torrent of titles. 2013. Disponível em: <http://www.publishersweekly.com>. Acesso em: 14 mar. 2015. Citado na página 26.

REFLECTZ. *Principais técnicas de levatamento de requisitos de sistemas*: Site. 2011. Disponível em: <https://brunobrum.wordpress.com/2011/04/27/principais-tecnicas-de-levantamento-de-requisitos-de-sistemas/>. Acesso em: 29 out. 2015. Citado na página 44.

REIS, H. C. *Especificação de um formato de arquivo open source para audiobooks com suporte a marcações de conteúdo*. 87 p. Monografia (Graduação) — Universidade de Brasília, 2013. Citado 7 vezes nas páginas 22, 49, 51, 54, 57, 67 e 72.

RENAULT, S. Pabre: Pattern-based requirements elicitation. IEEE, p. 81–92, 2009. Disponível em: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5089271&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D5089271>. Acesso em: 28 out. 2015. Citado na página 43.

RFC_3533. *The Ogg Encapsulation Format Version 0*: Documento técnico. 2003. Disponível em: <http://xiph.org/ogg/doc/rfc3533.txt>. Acesso em: 03 nov. 2014. Citado 6 vezes nas páginas 13, 36, 37, 38, 39 e 40.

RHJ. *Home page*: Site. 2009. Disponível em: <http://www.editorarhj.com.br>. Acesso em: 29 out. 2014. Citado na página 27.

SASSAKI, R. K. Terminologia sobre deficiência na era da inclusão. 2005. Disponível em: <https://acessibilidadecultural.wordpress.com/2011/09/07/terminologia-sobre-deficiencia-na-era-da-inclusao/>. Acesso em: 28 out. 2015. Citado na página 25.

SCIELO. *Scientific Electronic Library Online*: Site. 2014. Disponível em: <http://www.scielo.org/php/index.php>. Acesso em: 14 nov. 2014. Citado na página 47.

SERRA, F. *Áudio Digital: A tecnologia aplicada à música e ao tratamento de som*. [S.l.]: Editora Ciência Moderna Ltda, 2002. Citado na página 33.

SHAFFI, S.; WOOD, F. Accessibility is still a 'challenge' for publishers. 2014. Disponível em: <http://www.thebookseller.com>. Acesso em: 18 mar. 2015. Citado 2 vezes nas páginas 27 e 28.

SILVA, C. C. M.; TURATTO, J.; MACHADO, L. H. Os deficientes visuais e o acesso à informação. 2003. Disponível em: <http://www.ibc.gov.br/Nucleus/index.php?catid=4&itemid=46>. Acesso em: 21 mar. 2015. Citado na página 30.

SOFTWARE auxilia pessoas com deficiência visual: Site. 2012. Disponível em: <http://porvir.org/porcriar/software-auxilia-pessoas-deficiencia-visual/20120719>. Acesso em: 28 maio. 2015. Citado na página 30.

SOMMERVILLE, I. *Software Engineering*. Pearson/Addison-Wesley, 2004. (International computer science series). ISBN 9780321210265. Disponível em: <https://books.google.com.br/books?id=fIJQAAAAMAAJ>. Citado na página 43.

SOUZA, M. S. D.; CELVA, R. A.; HELVADJIAN, V. Audiolivro: um suporte para a educação literária. 2006. Disponível em: <http://ltp.emnuvens.com.br/ltp/article/viewFile/69/66>. Acesso em: 22 out. 2014. Citado 2 vezes nas páginas 26 e 27.

SOX. *Sound eXchande*: Site. 2014. Disponível em: <http://www.sox.sourceforge.net>. Acesso em: 20 set. 2014. Citado na página 53.

TEIXEIRA, J. Leitura de ouvido. Veja, p. 135, n. 35, ano 39, 2006. Citado na página 26.

WAGGENER, B. *Pulse Code Modulation Techniques*. [S.l.]: Springer, 1994. Citado na página 33.

XIPH.ORG. *Vorbis Audio Compression*: Site. 1994. Disponível em: <http://www.xiph.org/licenses/bsd/>. Acesso em: 15 nov. 2014. Citado na página 43.

XIPH.ORG. *Vorbis Audio Compression*: Site. 2000. Disponível em: <http://www.xiph.org/vorbis/>. Acesso em: 15 nov. 2014. Citado na página 43.

XIPH.ORG. *Vorbis I specification*: Site. 2012. Disponível em: <http://wiki.xiph.org/vorbis/doc/Vorbis_I_spec.html>. Acesso em: 03 nov. 2014. Citado 3 vezes nas páginas 41, 42 e 43.

Apêndices

APÊNDICE A – User Stories

Segue-se abaixo as User Stories descritas que define e prioriza os requisitos de acessibilidade para o Tocador. Aqui é possível identificar os atores, o cenário de possível interação com o utilizador do sistema, as funcionalidades e suas restrições, além dos pré-condições e pós-condições. Os critérios de aceitação para cada *User Story* estão descritos no Apêndice B.

Tabela 3: US01 - Interação por Teclado

Narrativa da Story	Interação por Teclado
Como	deficiente visual
Eu quero	manter o controle do tocador por meio do teclado
Para que	seja possível acessar e navegar pela informação contida no áudio.
Critérios de Aceitação	AC01, AC02, AC03, AC04, AC05, AC06, AC07, AC08, AC09, AC10, AC11, AC12, AC23 e AC24

Tabela 4: US02 - Sintetizador para Leitura de Tela

Narrativa da Story	Sintetizador para Leitura de Tela
Como	deficiente visual
Eu quero	ter uma resposta audível do que está na tela
Para que	saiba o que visualmente é mostrado da qual não consigo ver.
Critérios de Aceitação	Tabelas AC13 e AC14

Tabela 5: US03 - Contraste de tela

Narrativa da Story	Contraste de tela
Como	deficiente visual com baixa visão
Eu quero	poder alterar o contraste da tela
Para que	eu consiga enxergar a informação visual e fazer distinção entre o conteúdo apresentado.
Critérios de Aceitação	AC15 e AC25

Tabela 6: US04 - Sintetizador para informação de conteúdo

Narrativa da Story	Sintetizador para informação de conteúdo
Como	deficiente visual
Eu quero	ter uma resposta audível da informação do *audiobook*
Para que	eu tenha conhecimento do autor, ano de publicação, editora, número de páginas, título, endereço e idioma do livro que está sendo falado. Também quero ter conhecido da marcação e nível da marcação atual.
Critérios de Aceitação	AC16, AC17, AC23 e AC24

Tabela 7: US05 - Sintetizador para ajuda

Narrativa da Story	Sintetizador para ajuda
Como	deficiente visual
Eu quero	ter uma respota audível das funcionalidades da aplicação
Para que	eu seja ajudado quanto as funcionalidades e comandos disponíveis do Tocador.
Critérios de Aceitação	AC18 e AC19

Tabela 8: US06 - Destaque visual

Narrativa da Story	Destaque visual
Como	deficiente visual com baixa visão
Eu quero	gostaria que os botões sob o mouse fossem destacados
Para que	eu saiba onde está o cursor e também identifique o botão antes de apertá-lo.
Critérios de Aceitação	AC20

Tabela 9: US07 - Ampliador de tela

Narrativa da Story	Ampliador de tela
Como	deficiente visual com baixa visão
Eu quero	que os textos sob o mouse sejam ampliados
Para que	seja tornar possível ou facilitar a leitura.
Critérios de Aceitação	AC21

Tabela 10: US08 - Anúncio automático de botão sob o mouse

Narrativa da Story	Anúncio automático de botão sob o mouse
Como	deficiente visual com baixa visão
Eu quero	que ao mover o mouse o botão sob ele seja anunciado
Para que	eu saiba qual o botão antes mesmo de clicá-lo.
Critérios de Aceitação	AC22

APÊNDICE B – Critérios de Aceitação

Tabela 11: AC01 - Executar audiobook

Critério para Aceitação	Executar *Audiobook*
Dado	que o *audiobook* não esteja sendo executado
Quando	a tecla *space* for pressionada
Então	o *audiobook* deverá ser executado.
User Story	US01

Tabela 12: AC02 - Pausar audiobook

Critério para Aceitação	Pausar Execução do *Audiobook*
Dado	que o *audiobook* esteja sendo executado
Quando	a tecla *space* for pressionada
Então	a execução do *audiobook* deverá ser pausada.
User Story	US01

Tabela 13: AC03 - Ir para próxima marcação de mesmo nível

Critério para Aceitação	Ir para próxima marcação de mesmo nível
Dado	que o *audiobook* esteja sendo executado ou não
Quando	a tecla *direcional direita* for pressionada
Então	o Tocador deverá saltar para a próxima marcação de mesmo nível. Se for o final do *audiobook* a execução deverá ser pausada. Caso o *audiobook* esteja sendo executado ele deve continuar a execução a partir do salto. Caso o *audiobook* não esteja sendo executado ele deve continuar pausado para onde houve o salto.
User Story	US01

Tabela 14: AC04 - Ir para marcação anterior de mesmo nível

Critério para Aceitação	Ir para marcação anterior de mesmo nível
Dado	que o *audiobook* esteja sendo executado ou não
Quando	a tecla *direcional esquerda* for pressionada
Então	o Tocador deverá saltar para a marcação anterior de mesmo nível. Se for o início do *audiobook* a execução deverá continuar no estado atual a partir do tempo saltado. Caso o *audiobook* esteja sendo executado ele deve continuar a execução a partir do salto. Caso o *audiobook* não esteja sendo executado ele deve continuar pausado para onde houve o salto.
User Story	US01

Tabela 15: AC05 - Ir para próxima marcação

Critério para Aceitação	Ir para próxima marcação
Dado	que o *audiobook* esteja sendo executado ou não
Quando	a tecla *n* for pressionada
Então	o Tocador deverá saltar para a próxima marcação em relação ao tempo. Se for o final do *audiobook* a execução deverá ser pausada. Caso o *audiobook* esteja sendo executado ele deve continuar a execução a partir do salto. Caso o *audiobook* não esteja sendo executado ele deve continuar pausado para onde houve o salto.
User Story	US01

Tabela 16: AC06 - Ir para marcação anterior

Critério para Aceitação	Ir para marcação anterior
Dado	que o *audiobook* esteja sendo executado ou não
Quando	a tecla *f* for pressionada
Então	o Tocador deverá saltar para a marcação anterior em relação ao tempo. Se for o início do *audiobook* a execução deverá continuar no estado atual a partir do tempo saltado. Caso o *audiobook* esteja sendo executado ele deve continuar a execução a partir do salto. Caso o *audiobook* não esteja sendo executado ele deve continuar pausado para onde houve o salto.
User Story	US01

Tabela 17: AC07 - Subir nível

Critério para Aceitação	Subir nível
Dado	que o *audiobook* esteja sendo executado ou não
Quando	a tecla *direcional para cima* for pressionada
Então	o Tocador deverá alterar a navegação para um nível acima. O estado atual de execução deverá ser mantido.
User Story	US01

Tabela 18: AC08 - Descer nível

Critério para Aceitação	Descer nível
Dado	que o *audiobook* esteja sendo executado ou não
Quando	a tecla *direcional para baixo* for pressionada
Então	o Tocador deverá alterar a navegação para um nível abaixo. O estado atual de execução deverá ser mantido.
User Story	US01

Tabela 19: AC09 - Acessar metadados

Critério para Aceitação	Acessar metadados
Dado	que o *audiobook* esteja sendo executado ou não
Quando	a tecla *i* for pressionada
Então	o metadado deverá ser informado ao usuário. O estado atual de execução deverá ser mantido.
User Story	US01

Tabela 20: AC10 - Solicitar metadados

Critério para Aceitação	Solicitar ajuda
Dado	que o *audiobook* esteja sendo executado ou não
Quando	a tecla *h* for pressionada
Então	o usuário deverá ser informado sobre as teclas funcionais do Tocador. O estado atual de execução deverá ser mantido.
User Story	US01

Tabela 21: AC11 - Alterar contraste

Critério para Aceitação	Alterar contraste
Dado	que o *audiobook* esteja sendo executado ou não
Quando	a tecla *c* for pressionada
Então	o contraste de tela deverá ser alterado. O estado atual de execução deverá ser mantido.
User Story	US01

Tabela 22: AC12 - Tecla específica para cada botão

Critério para Aceitação	Tecla específica para cada botão
	Cada um dos botões do Tocador deve estar associado a um tecla específica.
User Story	US01

Tabela 23: AC13 - Resposta audível

Critério para Aceitação	Resposta audível
	Cada elemento presente na tela deve ter uma resposta audível descrevendo o que ele é.
User Story	US02

Tabela 24: AC14 - Informar atualização de texto em tela

Critério para Aceitação	Informar atualização de texto em tela
	Para cada atualização de texto em tela o usuário deverá ser informado de forma audível.
User Story	US02

Tabela 25: AC15 - Alternativas mínimas de contraste

Critério para Aceitação	Alternativas mínimas de contraste
	O Tocador deve ter no mínimo uma opção alternativa de contraste.
User Story	US03

Tabela 26: AC16 - Informação falada

Critério para Aceitação	Informação falada
	Para cada *audiobook* deve ser possível escutar o título, o autor, o idioma, a editora, a localidade, o número de páginas e o ano de publicação.
User Story	US04

Tabela 27: AC17 - Disponibilidade da informação

Critério para Aceitação	Disponibilidade da informação
	O acesso a esta informação deve estar disponível a qualquer momento que o usuário solicitar.
User Story	US04

Tabela 28: AC18 - Anúncio dos comandos

Critério para Aceitação	Anúncio dos comandos
	Cada funcionalidade disponível deve ser pronunciada.
User Story	US05

Tabela 29: AC19 - Anúncio dos comandos

Critério para Aceitação	Anúncio dos comandos
	Cada funcionalidade disponível deve ser pronunciada.
User Story	US05

Tabela 30: AC20 - Cor específica para botões

Critério para Aceitação	Cor específica para botões
	Cada botão deve ter uma cor específica.
User Story	US06

Tabela 31: AC21 - Ampliar texto sob mouse

Critério para Aceitação	Ampliar texto sob mouse
	Os textos sob o mouse devem ser automaticamente ampliados.
User Story	US07

Tabela 32: AC22 - Anunciar texto sob mouse

Critério para Aceitação	Anunciar botão sob mouse
	O botão sob o mouse deve ser anunciado imediatamente.
Dado	que o *audiobook* esteja sendo executado ou não
Quando	a mouse passar por cima do botão
Então	a descrição da funcionalidade do botão deverá ser falada ao usuário.
User Story	US08

Tabela 33: AC23 - Acessar marcação

Critério para Aceitação	Acessar marcação
Dado	que o *audiobook* esteja sendo executado ou não
Quando	a tecla *m* for pressionada
Então	a marcação atual deverá ser informada ao usuário. O estado atual de execução deverá ser mantido.
User Story	US01 e US04

Tabela 34: AC24 - Acessar nível da marcação

Critério para Aceitação	Acessar nível da marcação
Dado	que o *audiobook* esteja sendo executado ou não
Quando	a tecla *n* for pressionada
Então	o nível da marcação atual deverá ser informada ao usuário. O estado atual de execução deverá ser mantido.
User Story	US01 e US04

Tabela 35: AC25 - Alternar contraste de tela

Critério para Aceitação	Alternar contraste de tela
Dado	que o *audiobook* esteja sendo executado ou não
Quando	a tecla *c* for pressionada
Então	o contraste da tela deverá ser alternado entre o constrate padrão e o alternativo. O estado atual de execução deverá ser mantido.
User Story	US03